TMS470M Series
Technical Reference Manual (TRM)



Literature Number: SPNU495C
March 2013

Architecture Specification **17**
1.1 Overview **18**
1.2 Block Diagrams **19**
1.2.1 TMS470M Series with ARM Cortex M3 CPU 19
1.3 Memory Mapping **20**
1.3.1 Internal Program Memory 20
1.3.2 Endian System 26
1.4 System Module (SYS) **27**
1.5 Clock Definition **28**
1.5.1 Terminology 28
1.5.2 Clock Domains 28
1.5.3 Clock Domains and Low Power Mode 30
1.5.4 Primary/Secondary Clock Source Control 30
1.5.5 Oscillator Fail Detection 31
1.5.6 ECP 31
1.6 Resets **33**
1.7 Memory Ordering Model and Memory Protection **34**
1.7.1 Memory Attributes 34
1.7.2 Memory Types 34
1.7.3 Memory Ordering Model Variations 34
1.8 System Abort Concept and Illegal Transactions **36**
1.8.1 Abort Types 36
1.8.2 Accesses to Illegal Addresses 36
1.8.3 Illegal Accesses 37
1.8.4 Illegal Transaction Detection and Response 37
1.9 System Interrupts **39**
1.9.1 System Software Interrupt (SSI) 39
1.10 Memory Self Test **40**
1.10.1 MBIST 40
1.10.2 Memory BIST Implementation 40
1.10.3 Memory BIST Algorithms 40
1.11 Memory Module Hardware Initialization **42**
1.11.1 Memory Module Hardware Initialization Features 42

Low Power Mode Operation **43**
2.1 Overview **44**
2.2 Standard Power Management Functionality **44**
2.2.1 Overview 44
2.2.2 Active Clock Modes 44
2.2.3 Inactive Clock Modes 46
2.2.4 ARM CPU Idle Mechanisms 47

Interconnect **49**
3.1 Interconnect Overview **50**
3.2 Switched Central Resource (SCR) or Bus Matrix Module (BMM) **50**
3.3 Arbiter **50**
3.3.1 Arbitration Scheme 50
3.4 Peripheral Central Resource (PCR) **51**
3.4.1 PCR Operation 51
3.4.2 Clock Management Interface 52
3.4.3 Low Power Mode Interface 52

Control Registers **55**
4.1 System Peripherals Memory Mapping **56**

4.2 System Control Registers (SYS) 58
4.2.1 SYS Pin Control Register 1 (SYSPC1) 66
4.2.2 SYS Pin Control Register 2 (SYSPC2) 67
4.2.3 SYS Pin Control Register 3 (SYSPC3) 68
4.2.4 SYS Pin Control Register 4 (SYSPC4) 69
4.2.5 SYS Pin Control Register 5 (SYSPC5) 71
4.2.6 SYS Pin Control Register 6 (SYSPC6) 72
4.2.7 SYS Pin Control Register 7 (SYSPC7) 73
4.2.8 SYS Pin Control Register 8 (SYSPC8) 75
4.2.9 SYS Pin Control Register 9 (SYSPC9) 76
4.2.10 SSW PLL BIST Control Register 1 (SSWPLL1) 77
4.2.11 SSW PLL BIST Control Register 2 (SSWPLL2) 79
4.2.12 SSW PLL BIST Control Register3 (SSWPLL3) 80
4.2.13 Clock Source Disable Register (CSDIS) 81
4.2.14 Clock Source Disable Set Register (CSDISSET) 82
4.2.15 Clock Source Disable Clear Register (CSDISCLR) 83
4.2.16 Clock Domain Disable Register (CDDIS) 84
4.2.17 Clock Domain Disable Set Register (CDDISSET) 85
4.2.18 Clock Domain Disable Clear Register (CDDISCLR) 87
4.2.19 GCLK, HCLK, VCLK, and VCLK2 Source Register (GHVSRC) 89
4.2.20 Peripheral Asynchronous Clock Source Register (VCLKASRC) 91
4.2.21 RTI Clock Source Register (RCLKSRC) 93
4.2.22 Clock Source Valid Status Register (CSVSTAT) 95
4.2.23 Memory Self-Test Global Control Register (MSTGCR) 96
4.2.24 Memory Hardware Initialization Global Control Register (MINITGCR) 98
4.2.25 MBIST Controller/ Memory Initialization Enable Register (MSINENA) 99
4.2.26 Memory Self-Test Fail Status Register (MSTFAIL) 100
4.2.27 MSTC Global Status Register (MSTCGSTAT) 101
4.2.28 Memory Hardware Initialization Status Register (MINISTAT) 102
4.2.29 PLL Control Register 1 (PLLCTL1) 103
4.2.30 PLL Control Register 2 (PLLCTL2) 103
4.2.31 Die Identification Register Lower Word (DIEIDL) 104
4.2.32 Die Identification Register Upper Word (DIEIDH) 105
4.2.33 Voltage Regulator Control Register (VRCTL) 106
4.2.34 LPO/Clock Monitor Control Register (LPOMONCTL) 108
4.2.35 Clock Test Register (CLKTEST) 110
4.2.36 General Purpose Register 1(GPREG1) 113
4.2.37 BOOT ROM Select Register (BTRMSEL) 114
4.2.38 Imprecise Fault Status Register (IMPFASTS) 115
4.2.39 Imprecise Fault Address Register (IMPFTADD) 117
4.2.40 System Software Interrupt Request 1 Register (SSIR1) 118
4.2.41 System Software Interrupt Request 2 Register (SSIR2) 119
4.2.42 System Software Interrupt Request 3 Register (SSIR3) 120
4.2.43 System Software Interrupt Request 4 Register (SSIR4) 121
4.2.44 RAM Control Register (RAMGCR) 122
4.2.45 Bus Matrix Module Control Register1 (BMMCR1) 124
4.2.46 Bus Matrix Module Control Register2 (BMMCR2) 125
4.2.47 MMU Global Control Register (MMUGCR) 127
4.2.48 Clock Control Register (CLKCNTL) 128
4.2.49 ECP Control Register (ECPCNTL) 129
4.2.50 DSP Master Global Control Register (DSPGCR) 130
4.2.51 DEV Parity Control Register1 (DEVCR1) 131
4.2.52 System Exception Control Register (SYSECR) 132

4.2.53 System Exception Status Register (SYSESR) . . . 133
4.2.54 Global Status Register (GLBSTAT) . . . 135
4.2.55 Device Identification Register (DEVID) . . . 136
4.2.56 Software Interrupt Vector Register (SSIVEC) . . . 138
4.2.57 System Software Interrupt Flag Register (SSIF) . . . 139
4.3 Peripheral Central Resource (PCR) Control Registers . . . 140
4.3.1 Peripheral Memory Protection Set Register 0 (PMPROTSET0) . . . 148
4.3.2 Peripheral Memory Protection Set Register1 (PMPROTSET1) . . . 149
4.3.3 Peripheral Memory Protection Clear Register 0 (PMPROTCLR0) . . . 150
4.3.4 Peripheral Memory Protection Clear Register1 (PMPROTCLR1) . . . 151
4.3.5 Peripheral Protection Set Register 0 (PPROTSET0) . . . 152
4.3.6 Peripheral Protection Set Register 1 (PPROTSET1) . . . 154
4.3.7 Peripheral Protection Set Register 2 (PPROTSET2) . . . 155
4.3.8 Peripheral Protection Set Register 3 (PPROTSET3) . . . 156
4.3.9 Peripheral Protection Clear Register 0 (PPROTCLR0) . . . 157
4.3.10 Peripheral Protection Clear Register 1 (PPROTCLR1) . . . 158
4.3.11 Peripheral Protection Clear Register 2 (PPROTCLR2) . . . 159
4.3.12 Peripheral Protection Clear Register 3 (PPROTCLR3) . . . 160
4.3.13 Peripheral Memory Power-Down Set Register 0 (PCSPWRDWNSET0) . . . 161
4.3.14 Peripheral Memory Power-Down Set Register1 (PCSPWRDWNSET1) . . . 162
4.3.15 Peripheral Memory Power-Down Clear Register 0 (PCSPWRDWNCLR0) . . . 163
4.3.16 Peripheral Memory Power-Down Clear Register1 (PCSPWRDWNCLR1) . . . 164
4.3.17 Peripheral Power-Down Set Register 0 (PSPWRDWNSET0) . . . 165
4.3.18 Peripheral Power-Down Set Register 1 (PSPWRDWNSET1) . . . 167
4.3.19 Peripheral Power-Down Set Register 2 (PSPWRDWNSET2) . . . 168
4.3.20 Peripheral Power-Down Set Register 3 (PSPWRDWNSET3) . . . 169
4.3.21 Peripheral Power-Down Clear Register 0 (PSPWRDWNCLR0) . . . 170
4.3.22 Peripheral Power-Down Clear Register 1 (PSPWRDWNCLR1) . . . 171
4.3.23 Peripheral Power-Down Clear Register 2 (PSPWRDWNCLR2) . . . 172
4.3.24 Peripheral Power-Down Clear Register 3 (PSPWRDWNCLR3) . . . 173
4.4 System Control Registers Secondary Frame(SYS2) . . . 174
4.4.1 CPU Logic BIST Clock Divider (STCLKDIV) . . . 175
4.4.2 ECP Control Register (ECPCNTL1) . . . 176
4.4.3 ECP Control Register (ECPCNTL2) . . . 177

Embedded SRAM (eSRAM) . . . 179
5.1 General Description . . . 180
5.2 Block diagram . . . 181
5.3 Module operation . . . 184
5.4 Bit Access Operation . . . 185
5.5 Memory Fault Detection . . . 186
5.5.1 Read-Modify-Write Operation . . . 186
5.5.2 Consecutive Access . . . 187
5.5.3 ECC Memory Mapping . . . 188
5.5.4 ECC Generation . . . 188
5.5.5 Double Error Detection . . . 189
5.5.6 Single Error Correction . . . 190
5.5.7 False Double Error Detection . . . 190
5.5.8 Interrupt and Error Generation . . . 191
5.5.9 Emulation . . . 191
5.6 Hardware RAM Initialization . . . 192
5.7 Control Registers . . . 193
5.7.1 RAM Control Register (RAMCTRL) . . . 194

5.7.2 Threshold Register (RAMTHRESHOLD) . . . 196
5.7.3 Occurrence Register (RAMOCCUR) . . . 197
5.7.4 Interrupt Control Register (RAMINTCTRL) . . . 198
5.7.5 Memory Fault Detect Status Register (RAMERRSTATUS) . . . 199
5.7.6 Single Error Address Register (RAMSERRADD) . . . 200
5.7.7 RAM Error Position Register (RAMERRPOSITION) . . . 201
5.7.8 Double Error Address Register (RAMDERRADD) . . . 202
5.7.9 RAM Control Register (RAMCTRL2) . . . 203

Phase-Locked Loop (PLL) Clock Module . . . 205
6.1 Device Clock Overview . . . 206
6.2 FMzPLL Introduction/Feature Overview . . . 208
6.2.1 Features . . . 208
6.3 FMzPLL Operation . . . 209
6.3.1 Phase-Locked Loop (PLL) Description . . . 210
6.4 FMzPLL Control Registers . . . 215
6.4.1 Control registers . . . 215
6.5 FMzPLL Calculator (F035 FMzPLL Calculator) . . . 220
6.6 FMzPLL Configuration Example . . . 221

F035 Flash Module . . . 223
7.1 Overview . . . 224
7.1.1 Features . . . 224
7.1.2 Definition of Terms . . . 224
7.1.3 F035 Flash Tools . . . 225
7.2 Default flash configuration . . . 226
7.3 Memory Map . . . 227
7.3.1 Illegal Address Generation . . . 227
7.4 Operation . . . 228
7.4.1 Flash Read Modes . . . 228
7.4.2 Erase/Program Flash . . . 230
7.4.3 ECC Protection . . . 231
7.4.4 Data Security . . . 239
7.4.5 Automatic Power-down of Flash Banks . . . 240
7.5 Control Registers . . . 241
7.5.1 Flash Option Control Register (FRDCNTL) . . . 245
7.5.2 Flash Special Read Control Register (FSPRD) . . . 246
7.5.3 Flash Error Detection and Correction Control Register 1 (FEDACCTRL1 - 0xFFF87008) . . . 247
7.5.4 Flash Error Correction and Correction Control Register 2 (FEDACCTRL2 - 0xFFF8700C) . . . 250
7.5.5 Flash Error Correction Counter Register (FCOR_ERR_CNT - 0xFFF87010) . . . 251
7.5.6 Flash Correctable Error Address (FCOR_ERR_ADD - 0xFFF87014) . . . 252
7.5.7 Correctable Error Position Register (FCOR_ERR_POS) . . . 253
7.5.8 Flash Error Status Register (FEDACSTATUS - 0xFFF8701C) . . . 255
7.5.9 Flash Un-correctable Error Address (FUNC_ERR_ADD - 0xFFF87020) . . . 257
7.5.10 Flash Error Detection Sector Disable (FEDACSDIS - 0xFFF87024) . . . 258
7.5.11 Flash Bank Protection Register (FBPROT - 0xFFF87030) . . . 259
7.5.12 Flash Bank Sector Enable Register (FBSE- 0xFFF87034) . . . 260
7.5.13 Flash Bank Access Control Register (FBAC - 0x3C) . . . 261
7.5.14 Flash Bank Fallback Power Register (FBFALLBACK - 0xFFF87040) . . . 262
7.5.15 Flash Bank/Pump Ready Register (FBPRDY - 0xFFF87044) . . . 263
7.5.16 Flash Pump Access Control Register 1 (FPAC1 - 0xFFF87048) . . . 264
7.5.17 Flash Pump Access Control Register 2 (FPAC2 - 0xFFF8704C) . . . 265
7.5.18 Flash Module Access Control Register (FMAC - 0xFFF87050) . . . 266
7.5.19 Flash Emulation ECC Register (FEMU_ECC - 0xFFF87060) . . . 269

7.5.20 Flash Error Detection Sector Disable (FEDACSDIS2 - 0xFFF870C0) 270

Cortex-M3 Vectored Interrupt Manager (M3VIM) Module 271
8.1 Overview 272
8.1.1 Interrupt Handling at the CPU 272
8.1.2 Nesting Behavior 272
8.1.3 Interrupt Generation at the Peripheral 273
8.2 Interrupt management 274
8.2.1 M3VIM Interrupt request management 274
8.2.2 M3VIM Wake-up Interrupt 276
8.2.3 M3VIM Input Channel Management 276
8.2.4 M3VIM Prioritization 277
8.3 M3VIM Operation 279
8.3.1 Vector Table 279
8.3.2 M3VIM response to interrupt request 280
8.3.3 .Emulation 280
8.4 Capture event sources 281
8.5 Programmer's Model Notes 282
8.5.1 Required Software Initialization 282
8.5.2 NVIC vs. M3VIM controls 282
8.5.3 Autovectoring versus INTNMI Interrupts 283
8.5.4 Interrupt Clearing 283
8.5.5 Reset 283
8.6 Registers 284
8.6.1 M3VIM Channel Offset Registers 290
8.6.2 NESTCTRL Register 293
8.6.3 NESTSTAT Register 294
8.6.4 INTNMI/INTISR Program Control Registers (NMIPRx) 296
8.6.5 Pending Interrupt Read Location Registers (INTREQx) 299
8.6.6 Interrupt Mask Set Registers (REQMASKSETx) 302
8.6.7 Interrupt Mask Clear Registers (REQMASKCLRx) 305
8.6.8 Wake-up Mask Set Registers (WAKEMASKSETx) 308
8.6.9 Wake-up Mask Clear Registers (WAKEMASKCLRx) 311
8.6.10 Capture Event Register (CAPEVT) 314
8.6.11 M3VIM Interrupt Control Register x (CHANCTRL x) 315

Cyclic Redundancy Check Controller (CRC) Module 319
9.1 Overview 320
9.2 TMS470M Series CRC Features 321
9.3 Module Operation 322
9.3.1 General Operation 322
9.3.2 Modes 322
9.3.3 Register Definitions 322
9.3.4 Power-Down Mode 323
9.3.5 Emulation 324
9.4 CRC Control Registers 325
9.4.1 CRC Global Control Register 0 (CRC_CTRL0) 327
9.4.2 CRC Global Control Register (CRC_CTRL1) 328
9.4.3 CRC Global Control Register 2 (CRC_CTRL2) 329
9.4.4 PSA Signature Low Register 1 (PSA_SIGREGL1) 331
9.4.5 PSA Signature High Register 1 (PSA_SIGREGH1) 332
9.4.6 PSA Sector Signature Low Register 1 (PSA_SECSIGREGL1) 333
9.4.7 PSA Sector Signature High Register 1 (PSA_SECSIGREGH1) 334
9.4.8 Raw Data Low Register 1 (RAW_DATAREGL1) 335

9.4.9 Raw Data High Register 1 (RAW_DATAREGH1) 336

Controller Area Network (DCAN) 337
10.1 Overview 338
10.1.1 Features 338
10.1.2 Functional Description 338
10.1.3 Block Diagram 339
10.2 Operating Modes 341
10.2.1 Software Initialization 341
10.2.2 CAN Message Transfer (Normal Operation) 341
10.2.3 Test Modes 343
10.3 Dual Clock Source 347
10.4 GIO support 348
10.5 RAM Initialization 349
10.6 Interrupt functionality 350
10.6.1 Message Object interrupts 350
10.6.2 Status Change Interrupts 350
10.6.3 Error Interrupts 350
10.7 Global power down mode 351
10.7.1 Entering global power down mode 351
10.7.2 Wakeup from global power down mode 351
10.8 Local power down mode 352
10.8.1 Entering local power down mode 352
10.8.2 Wakeup from local power down 352
10.9 Parity Check Mechanism 354
10.9.1 Behavior on parity error 354
10.9.2 Parity testing 354
10.10 Debug/Suspend Mode 355
10.11 Module Initialization 356
10.12 Configuration of Message Objects 357
10.12.1 Configuration of a Transmit Object for Data Frames 358
10.12.2 Configuration of a Transmit Object for Remote Frames 358
10.12.3 Configuration of a Single Receive Object for Data Frames 358
10.12.4 Configuration of a Single Receive Object for Remote Frames 359
10.12.5 Configuration of a FIFO Buffer 359
10.13 Message Handling 360
10.13.1 Message Handler Overview 360
10.13.2 Receive/Transmit Priority 361
10.13.3 Transmission of Messages in Event Driven CAN Communication 361
10.13.4 Updating a Transmit Object 361
10.13.5 Changing a Transmit Object 361
10.13.6 Acceptance Filtering of Received Messages 362
10.13.7 Reception of Data Frames 362
10.13.8 Reception of Remote Frames 362
10.13.9 Reading Received Messages 363
10.13.10 Requesting New Data for a Receive Object 363
10.13.11 Storing Received Messages in FIFO Buffers 363
10.13.12 Reading from a FIFO Buffer 363
10.14 CAN Bit Timing 365
10.14.1 Bit Time and Bit Rate 365
10.14.2 Configuration of the DCAN Bit Timing 371
10.15 Message Interface Register Sets 374
10.15.1 Message Interface Register Sets 1 and 2 375

10.15.2 IF3 Register Set 376
10.16 Message RAM. 377
10.16.1 Structure of Message Objects 378
10.16.2 Addressing Message Objects in RAM 382
10.16.3 Message RAM representation in Debug/Suspend Mode 383
10.16.4 Message RAM representation in Direct Access Mode 383
10.17 DCAN Control Registers 385
10.17.1 CAN Control Register (DCAN CTL) 392
10.17.2 Error and Status Register (DCAN ES) 396
10.17.3 Error Counter Register (DCAN ERRC). 399
10.17.4 Bit Timing Register (DCAN BTR) 400
10.17.5 Interrupt Register (DCAN INT) 402
10.17.6 Test Register (DCAN TEST) 404
10.17.7 Parity Error Code Register (DCAN PERR). 406
10.17.8 DCAN Core Release Register (DCAN REL) 407
10.17.9 Auto-Bus-On Time Register (DCAN ABOTR) 408
10.17.10Transmission Request X Register (DCAN TXRQ X). 409
10.17.11Transmission Request Registers (DCAN TXRQ12 to DCAN TXRQ78) 410
10.17.12New Data X Register (DCAN NWDAT X) 411
10.17.13New Data Registers (DCAN NWDAT12 to DCAN NWDAT78) 412
10.17.14Interrupt Pending X Register (DCAN INTPND X) 413
10.17.15Interrupt Pending Registers (DCAN INTPND12 to DCAN INTPND78) 414
10.17.16Message Valid X Register (DCAN MSGVAL X) 414
10.17.17Message Valid Registers (DCAN MSGVAL12 to DCAN MSGVAL78) 416
10.17.18Interrupt Multiplexer Registers (DCAN INTMUX12 to DCAN INTMUX78) 416
10.17.19IF1/IF2 Command Registers (DCAN IF1CMD, DCAN IF2CMD). 417
10.17.20IF1/IF2 Mask Registers (DCAN IF1MSK, DCAN IF2MSK) 422
10.17.21IF1/IF2 Arbitration Registers (DCAN IF1ARB, DCAN IF2ARB) 424
10.17.22IF1/IF2 Message Control Registers (DCAN IF1MCTL, DCAN IF2MCTL). 426
10.17.23IF1/IF2 Data A and Data B Registers (DCAN IF1DATA/DATB, DCAN IF2DATA/DATB) 429
10.17.24IF3 Observation Register (DCAN IF3OBS). 431
10.17.25IF3 Mask Register (DCAN IF3MSK) 434
10.17.26IF3 Arbitration Register (DCAN IF3ARB) 435
10.17.27IF3 Message Control Register (DCAN IF3MCTL). 436
10.17.28IF3 Data A and Data B Registers (DCAN IF3DATA/DATB) 438
10.17.29IF3 Update Enable Registers (DCAN IF3UPD12 to IF3UPD78). 439
10.17.30CAN TX IO Control Register (DCAN TIOC) 440
10.17.31CAN RX IO Control Register (DCAN RIOC) 442

Multi-Buffered Serial Peripheral Interface Module (MibSPI) with Parallel Pin Option (MibSPIP) 445
11.1 Overview 446
11.1.1 Word Format Options 446
11.1.2 Multi-buffering (Mib) support 446
11.1.3 Transmission Lock (Multi-Buffer Mode Master Only) 447
11.2 Operating Modes 448
11.2.1 Pin Configurations 448
11.2.2 Data Handling. 448
11.2.3 Operation with SPISCS 450
11.2.4 Operation with SPIENA 451
11.2.5 Five-Pin Operation (Hardware Handshaking). 452
11.2.6 Data Formats 453
11.2.7 Clocking Modes 454
11.2.8 Data Transfer Example 456

11.2.9 Decoded and Encoded Chip Select (Master Only) . . . 457
11.2.10 Variable Chip Select Setup and Hold Timing (Master Only) . . . 457
11.2.11 Hold Chip-Select Active . . . 457
11.2.12 Detection of Slave Desynchronization (Master Only) . . . 458
11.2.13 ENA Signal Time-Out (Master Only) . . . 458
11.2.14 Data-Length Error . . . 459
11.2.15 Parallel Mode (Multiple SIMO/SOMI Support, not available on all devices) . . . 459
11.2.16 Continuous Self-Test (Master/Slave) . . . 466
11.3 Test Features . . . 467
11.3.1 Internal Loop-Back Test Mode (Master Only) . . . 467
11.3.2 Input/Output Loopback Test Mode . . . 467
11.4 General-Purpose I/O . . . 469
11.5 Low-Power Mode . . . 470
11.6 Interrupts . . . 471
11.6.1 Interrupts in Multi-Buffer Mode . . . 471
11.7 DMA Interface . . . 474
11.7.1 DMA in Multi-Buffer Mode . . . 474
11.8 Module Configuration . . . 475
11.8.1 Compatibility(SPI) Mode Configuration . . . 475
11.8.2 MibSPI Mode Configuration . . . 475
11.9 Control Registers . . . 477
11.9.1 SPI Global Control Register 0 (SPIGCR0) . . . 483
11.9.2 SPI Global Control Register 1 (SPIGCR1) . . . 484
11.9.3 SPI Interrupt Register (SPIINT0) . . . 486
11.9.4 SPI Interrupt Level Register (SPILVL) . . . 489
11.9.5 SPI Flag Register (SPIFLG) . . . 491
11.9.6 SPI Pin Control Register 0 (SPIPC0) . . . 496
11.9.7 SPI Pin Control Register 1 (SPIPC1) . . . 498
11.9.8 SPI Pin Control Register 2 (SPIPC2) . . . 500
11.9.9 SPI Pin Control Register 3 (SPIPC3) . . . 502
11.9.10 SPI Pin Control Register 4 (SPIPC4) . . . 504
11.9.11 SPI Pin Control Register 5 (SPIPC5) . . . 506
11.9.12 SPI Pin Control Register 6 (SPIPC6) . . . 508
11.9.13 SPI Pin Control Register 7(SPIPC7) . . . 510
11.9.14 SPI Pin Control Register 8(SPIPC8) . . . 512
11.9.15 SPI Transmit Data Register 0 (SPIDAT0) . . . 514
11.9.16 SPI Transmit Data Register 1 (SPIDAT1) . . . 515
11.9.17 SPI Receive Buffer Register (SPIBUF) . . . 517
11.9.18 SPI Emulation Register (SPIEMU) . . . 521
11.9.19 SPI Delay Register (SPIDELAY) . . . 522
11.9.20 SPI Default Chip Select Register (SPIDEF) . . . 526
11.9.21 SPI Data Format Registers (SPIFMT[3:0]) . . . 527
11.9.22 Interrupt Vector 0 (INTVECT0) . . . 530
11.9.23 Interrupt Vector 1 (INTVECT1) . . . 532
11.9.24 Parallel/Modulo Mode Control Register (SPIPMCTRL) . . . 534
11.9.25 Multi-buffer Mode Enable Register (MIBSPIE) . . . 538
11.9.26 TG Interrupt Enable Set Register (TGITENST) . . . 540
11.9.27 MibSPI TG Interrupt Enable Clear Register (TGITENCR) . . . 541
11.9.28 Transfer Group Interrupt Level Set Register (TGITLVST) . . . 542
11.9.29 Transfer Group Interrupt Level Clear Register (TGITLVCR) . . . 543
11.9.30 Transfer Group Interrupt Flag Register (TGINTFLAG) . . . 544
11.9.31 Tick Count Register (TICKCNT) . . . 546
11.9.32 Last TG End Pointer (LTGPEND) . . . 548

11.9.33 TGx Control Registers (TGxCTRL) . . . 549
11.9.34 DMA Channel Control Register (DMAxCTRL) . . . 555
11.9.35 DMAxCOUNT Register (ICOUNT) . . . 559
11.9.36 DMA Large Count (DMACNTLEN) . . . 560
11.9.37 Multi-buffer RAM Uncorrectable Parity Error Control Register (UERRCTRL) . . . 561
11.9.38 Multi-buffer RAM Uncorrectable Parity Error Status Register (UERRSTAT) . . . 562
11.9.39 RXRAM Uncorrectable Parity Error Address Register (UERRADDR1) . . . 563
11.9.40 TXRAM Uncorrectable Parity Error Address Register (UERRADDR0) . . . 564
11.9.41 RXRAM Overrun Buffer Address Register (RXOVRN_BUF_ADDR) . . . 565
11.9.42 I/O-Loopback Test Control Register (IOLPBKTSTCR) . . . 566
11.10 Multi-Buffer RAM . . . 569
11.10.1 Multi-Buffer RAM Auto Initialization . . . 570
11.10.2 Multi-buffer RAM Register Summary . . . 571
11.10.3 Multi-buffer RAM Transmit Data Register . . . 572
11.10.4 Multi-buffer RAM Receive Buffer Register . . . 575
11.11 Parity Memory . . . 579
11.11.1 Example of Parity Memory Organization . . . 581
11.12 MibSPI Pin Timing Parameters . . . 582
11.12.1 Master Mode Timings for SPI/MibSPI . . . 582
11.12.2 Slave Mode Timings for SPI/MibSPI . . . 583
11.12.3 Timing Parameters of SPI/MibSPI pins in all the modes . . . 585

High-End Timer w/Parity (HET) Module . . . 587
12.1 Features . . . 588
12.2 Overview . . . 589
12.2.1 Timer Module Structure and Execution . . . 589
12.2.2 Major Advantages . . . 590
12.2.3 Performance . . . 590
12.2.4 Instructions Features . . . 590
12.2.5 Parity Support . . . 591
12.2.6 Block Diagram . . . 591
12.3 HET Functional Description . . . 593
12.3.1 Host Interface . . . 593
12.3.2 HET RAM . . . 596
12.3.3 Time Base . . . 597
12.3.4 Specialized Timer Micromachine . . . 600
12.3.5 I/O Control . . . 605
12.3.6 Interrupts and Exceptions . . . 616
12.3.7 Hardware Priority Scheme . . . 617
12.4 HET Parity . . . 619
12.5 Angle Functions . . . 620
12.5.1 Software Angle Generator (SWAG) . . . 620
12.6 HET Control Registers . . . 624
12.6.1 Global Configuration Register (HETGCR) . . . 628
12.6.2 Prescale Factor Register (HETPFR) . . . 630
12.6.3 HET Current Address Register (HETADDR) . . . 632
12.6.4 Offset Index Priority Level 1 Register (HETOFF1) . . . 633
12.6.5 Offset Index Priority Level 2 Register (HETOFF2) . . . 634
12.6.6 Exception Control Register 1 (HETEXC1) . . . 635
12.6.7 Exception Control Register 2 (HETEXC2) . . . 636
12.6.8 Interrupt Priority Register (HETPRY) . . . 637
12.6.9 HET Interrupt Flag Register (HETFLG) . . . 638
12.6.10 HR Share Control Register (HETHRSH) . . . 639

12.6.11 HR XOR-Share Control Register (HETXOR) . . . 640
12.6.12 HET Direction Register (HETDIR) . . . 641
12.6.13 HET Data Input Register (HETDIN) . . . 642
12.6.14 HET Data Output Register (R-Write) (HETDOUT) . . . 643
12.6.15 HET Data Set Register (R-Set) (HETDSET) . . . 644
12.6.16 HET Data Clear Register (R-Clear) (HETDCLR) . . . 645
12.6.17 HET Open Drain Register (HETPDR) . . . 646
12.6.18 HET Pull Disable Register (HETPULDIS) . . . 647
12.6.19 HET Pull Select Register (HETPSL) . . . 648
12.6.20 HET Loopback Pair Select Register (HETLPBSEL) . . . 649
12.6.21 HET Loopback Pair Direction Register (HETLPBDIR) . . . 650
12.6.22 HET Parity Control Register (HETPCR) . . . 651
12.6.23 HET Parity Interrupt Enable Register (HETPIEN) . . . 653
12.6.24 HET Parity Interrupt Flag Register (HETPIFLG) . . . 654
12.6.25 HET Parity Address Register (HETPAR) . . . 655
12.7 Instruction Set . . . 656
12.7.1 Abbreviations . . . 658
12.7.2 Encoding Formats and Bits . . . 658
12.7.3 Instruction Description . . . 661

Cortex M3 LOCKUP Reset Module (LRM) . . . 715
13.1 Overview . . . 716
13.2 Features . . . 716
13.2.1 LOCKUP Pulse Generation . . . 716
13.2.2 LOCKUP Watchdog Reset . . . 716

CPU Self Test Controller (LBIST) Module . . . 717
14.1 General Description . . . 718
14.2 Deterministic Logic BIST concept . . . 719
14.3 STC Block diagram . . . 721
14.4 Module Description . . . 723
14.4.1 ROM Interface . . . 723
14.4.2 FSM and Sequence control . . . 723
14.4.3 Register Block . . . 723
14.4.4 STC Bypass / ATE Interface . . . 723
14.4.5 VBUSP Interface . . . 723
14.5 Application Self Test Flow Chart . . . 724
14.6 SelfTest Execution Flow . . . 725
14.7 Self Test Completion and Error Generation . . . 726
14.8 STC clock Divider . . . 727
14.9 Control Registers . . . 728
14.9.1 Suspend mode consideration: . . . 729
14.9.2 STC global control register0 (STCGCR0) . . . 730
14.9.3 STC Global Control Register1 (STCGCR1) . . . 731
14.9.4 Self Test Run Timeout Counter Preload Register (STCTPR) . . . 732
14.9.5 STC Current ROM Address Register (STC_CADDR) . . . 733
14.9.6 STC Current Interval Count Register (STCCICR) . . . 734
14.9.7 SelfTest Global Status Register (STCGSTAT) . . . 735
14.9.8 SelfTest Fail Status Register (STCFSTAT) . . . 736
14.9.9 CPU1 Current MISR Register (CPU1_CURMISR[3:0]) . . . 737
14.10 ROM Organization . . . 739
14.11 Timing Diagrams . . . 741
14.11.1 BIST Operation: For a self test run for One Interval . . . 741

Analog To Digital Converter (ADC) Module 743
15.1 Overview 744
15.2 Introduction 745
15.2.1 Analog Input Multiplexer 746
15.2.2 Self-Test and Calibration 746
15.2.3 Analog-to-Digital Converter Core 746
15.2.4 Sequencer 746
15.2.5 Conversion groups 746
15.3 Basic Features and Usage of the ADC 747
15.3.1 How to setup the ADCLK speed and the acquisition time? 749
15.3.2 How to initialize ADC Results FIFO RAM? 750
15.3.3 How to select an input channel for conversion? 750
15.3.4 How to configure single or continuous modes? 750
15.3.5 How to configure software or hardware trigger? 750
15.3.6 How to start a conversion? 750
15.3.7 How to know the group conversion is completed? 751
15.3.8 How are results stored in the results memory? 751
15.3.9 How to read the results from the results memory? 752
15.3.10 How to stop a conversion? 754
15.3.11 List of ‘Do NOT’ 754
15.4 Advanced Conversion Group Configuration Options 755
15.4.1 Single or Continuous Conversion Modes 755
15.4.2 Conversion Group Freeze Capability 755
15.4.3 Conversion Group Priority 756
15.4.4 8-bit or 10-bit Result Mode 756
15.4.5 Group Memory Overrun Option 756
15.4.6 Group Channel Id Storage Option 756
15.4.7 Group Trigger Options 756
15.5 ADC Module Interrupts 758
15.5.1 Group Conversion End Interrupt 758
15.5.2 Group Memory Threshold Interrupt 758
15.5.3 Group Memory Overrun Interrupt 758
15.5.4 ADC Magnitude Threshold Interrupts 758
15.6 ADC Error Calibration 761
15.6.1 Calibration and Offset Error Correction Procedure 761
15.6.2 Mid-Point Calibration 762
15.7 ADC Built In Diagnostics and Self Test Logic 764
15.7.1 ADC RAM Parity and Test 764
15.7.2 ADC Self-Test Mode 765
15.8 ADC Special Modes 768
15.8.1 ADC Powerdown Mode 768
15.8.2 ADC Sample Capacitor Discharge Mode 768
15.9 ADEVT Pin General Purpose I/O Functionality 769
15.9.1 GPIO Functionality 769
15.9.2 Summary 769
15.10 ADC Control Registers 771
15.10.1 ADC Reset Control Register (ADRSTCR) 781
15.10.2 ADC Operating Mode Control Register (ADOPMODECR) 782
15.10.3 ADC Clock Control Register (ADCLOCKCR) 784
15.10.4 ADC Calibration Mode Control Register (ADCALCR) 785
15.10.5 ADC Event Group Operating Mode Control Register (ADEVMODECR) 787
15.10.6 ADC Group1 Operating Mode Control Register (ADG1MODECR) 789
15.10.7 ADC Group2 Operating Mode Control Register (ADG2MODECR) 791

15.10.8 ADC Trigger Source Select Register (ADEVSRC, ADG1SRC and ADG2SRC) 794
15.10.9 ADC Event Interrupt Enable Control Register (ADEVINTENA) 796
15.10.10ADC Group1 Interrupt Enable Control Register (ADG1INTENA) 797
15.10.11ADC Group2 Interrupt Enable Control Register (ADG2INTENA) 798
15.10.12ADC Event Group Interrupt Flag Register (ADEVINTFLG) 799
15.10.13ADC Group1 Interrupt Flag Register (ADG1INTFLG) 801
15.10.14ADC Group2 Interrupt Flag Register (ADG2INTFLG) 803
15.10.15ADC Event Group Threshold Interrupt Control Register (ADEVTHRINTCR) 805
15.10.16ADC Group1 Threshold Interrupt Control Register (ADG1THRINTCR) 806
15.10.17ADC Group2 Threshold Interrupt Control Register (ADG2THRINTCR) 807
15.10.18ADC Results Memory Configuration Register (ADBNDCR) 808
15.10.19ADC Results Memory Size Configuration Register (ADBNDEND) 809
15.10.20ADC Event Group Sampling Time Configuration Register (ADEVSAMP) 811
15.10.21ADC Group1 Sampling Time Configuration Register (ADG1SAMP) 812
15.10.22ADC Group2 Sampling Time Configuration Register (ADG2SAMP) 813
15.10.23ADC Event Group Status Register (ADEVSR) 814
15.10.24ADC Group1 Status Register (ADG1SR) 816
15.10.25ADC Group2 Status Register (ADG2SR) 818
15.10.26ADC Event Group Channel Select Register (ADEVSEL) 820
15.10.27ADC Group1 Channel Select Register (ADG1SEL) 821
15.10.28ADC Group2 Channel Select Register (ADG2SEL) 822
15.10.29ADC Calibration and Error Offset Correction Register (ADCALR) 823
15.10.30ADC Channel Last Conversion Value Register (ADLASTCONV) 824
15.10.31ADC Event Group Results FIFO (ADEVBUFFER) 825
15.10.32ADC Group1 Results FIFO (ADG1BUFFER) 826
15.10.33ADC Group2 Results FIFO (ADG2BUFFER) 827
15.10.34ADC Event Group Results Emulation FIFO (ADEVEMUBUFFER) [offset = 0xF0] 828
15.10.35ADC Group1 Results Emulation FIFO (ADG1EMUBUFFER) [offset = 0xF4] 828
15.10.36ADC Group2 Results Emulation FIFO (ADG2EMUBUFFER) [offset = 0xF8] 828
15.10.37ADC ADEVT Pin Direction Control Register (ADEVTDIR) 829
15.10.38ADC ADEVT Pin Output Value Control Register (ADEVTOUT) 830
15.10.39ADC ADEVT Pin Input Value Register (ADEVTIN) 831
15.10.40ADC ADEVT Pin Set Register (ADEVTSET) 832
15.10.41ADC ADEVT Pin Clear Register (ADEVTCLR) 833
15.10.42ADC ADEVT Pin Open Drain Enable Register (ADEVTPDR) 834
15.10.43ADC ADEVT Pin Pull Control Disable Register (ADEVTPDIS) 835
15.10.44ADC ADEVT Pin Pull Control Select Register (ADEVTPSEL) 836
15.10.45ADC Event Group Sample Cap Discharge Control Register (ADEVSAMPDISEN) 837
15.10.46ADC Group1 Sample Cap Discharge Control Register (ADG1SAMPDISEN) 838
15.10.47ADC Group2 Sample Cap Discharge Control Register (ADG2SAMPDISEN) 839
15.10.48ADC Magnitude Compare Interrupt Control Registers (ADMAGINTxCR) 840
15.10.49ADC Magnitude Compare Mask (ADMAGxMASK) 842
15.10.50ADC Magnitude Compare Interrupt Enable Set (ADMAGINTENASET) 843
15.10.51ADC Magnitude Compare Interrupt Enable Clear (ADMAGINTENACLR) 844
15.10.52ADC Magnitude Compare Interrupt Flag (ADMAGINTFLG) 845
15.10.53ADC Magnitude Compare Interrupt Offset (ADMAGINTOFF) 846
15.10.54ADC Event Group FIFO Reset Control Register (ADEVFIFORESETCR) 847
15.10.55ADC Group1 FIFO Reset Control Register (ADG1FIFORESETCR) 848
15.10.56ADC Group2 FIFO Reset Control Register (ADG2FIFORESETCR) 849
15.10.57ADC Event Group RAM Write Address (ADEVRAMWRADDR) 850
15.10.58ADC Group1 RAM Write Address (ADG1RAMWRADDR) 851
15.10.59ADC Group2 RAM Write Address (ADG2RAMWRADDR) 852
15.10.60ADC Parity Control Register (ADPARCR) 853

15.10.61ADC Parity Error Address (ADPARADDR) 854

Real-Time Interrupt (RTI) Module 855
16.1 Introduction and Feature Overview 856
16.1.1 Purpose 856
16.1.2 Main Features 856
16.1.3 Industry Standard Compliance Statement 856
16.2 Module Operation 857
16.2.1 Counter Operation 857
16.2.2 Clock Domain 858
16.2.3 Digital Watchdog (DWD) 859
16.3 Control Registers 861
16.3.1 RTI Global Control Register (RTIGCTRL) 865
16.3.2 RTI Capture Control Register (RTICAPCTRL) 866
16.3.3 RTI Compare Control Register (RTICOMPCTRL) 867
16.3.4 RTI Free Running Counter 0 Register (RTIFRC0) 868
16.3.5 RTI Up Counter 0 Register (RTIUC0) 869
16.3.6 RTI Compare Up Counter 0 Register (RTICPUC0) 870
16.3.7 RTI Capture Free Running Counter 0 Register (RTICAFRC0) 871
16.3.8 RTI Capture Up Counter 0 Register (RTICAUC0) 872
16.3.9 RTI Free Running Counter 1 Register (RTIFRC1) 873
16.3.10 RTI Up Counter 1 Register (RTIUC1) 874
16.3.11 RTI Compare Up Counter 1 Register (RTICPUC1) 875
16.3.12 RTI Capture Free Running Counter 1 Register (RTICAFRC1) 876
16.3.13 RTI Capture Up Counter 1 Register (RTICAUC1) 877
16.3.14 RTI Compare 0 Register (RTICOMP0) 878
16.3.15 RTI Update Compare 0 Register (RTIUDCP0) 879
16.3.16 RTI Compare 1 Register (RTICOMP1) 880
16.3.17 RTI Update Compare 1 Register (RTIUDCP1) 881
16.3.18 RTI Compare 2 Register (RTICOMP2) 882
16.3.19 RTI Update Compare 2 Register (RTIUDCP2) 883
16.3.20 RTI Compare 3 Register (RTICOMP3) 884
16.3.21 RTI Update Compare 3 Register (RTIUDCP3) 885
16.3.22 RTI Set/Status Interrupt Register (RTISETINT) 886
16.3.23 RTI Clear/Status Interrupt Register (RTICLEARINT) 888
16.3.24 RTI Interrupt Flag Register (RTIINTFLAG) 890
16.3.25 Digital Watchdog Control Register (RTIDWDCTRL) 892
16.3.26 RTI Digital Watchdog Preload Register (RTIDWDPRLD) 893
16.3.27 Watchdog Status Register (RTIWDSTATUS) 894
16.3.28 RTI Watchdog Key Register (RTIWDKEY) 895
16.3.29 RTI Digital Watchdog Down Counter (RTIDWDCNTR) 896

Error Signaling (ESM) Module 897
17.1 General Description 898
17.1.1 Feature List 898
17.2 Block diagram 899
17.3 Register Mapping 900
17.4 Module Operation 901
17.5 Control Registers 903
17.5.1 Control registers 903
17.5.2 ESM Influence Error Pin Set/Status Register 1 (ESMIEPSR1) 904
17.5.3 ESM Influence Error Pin Clear/Status Register 1 (ESMIEPCR1) 906
17.5.4 ESM Interrupt Enable Set/Status Register 1 (ESMIESR1) 907
17.5.5 ESM Interrupt Enable Clear/Status Register 1 (ESMIECR1) 908

17.5.6 ESM Interrupt Level Set/Status Register 1 (ESMILSR1) . . . 909
17.5.7 ESM Interrupt Level Clear/Status Register 1 (ESMILCR1) . . . 910
17.5.8 ESM Status Register 1 (ESMSR1) . . . 911
17.5.9 ESM Status Register 2 (ESMSR2) . . . 912
17.5.10 ESM Status Register 3 (ESMSR3) . . . 913
17.5.11 ESM Error Pin Status Register (ESMEPSR) . . . 914
17.5.12 ESM Interrupt Offset High Register (ESMIOFFHR) . . . 915
17.5.13 ESM Interrupt Offset Low Register (ESMIOFFLR) . . . 916
17.5.14 ESM Low-Time Counter Register (ESMLTCR) . . . 917
17.5.15 ESM Low-Time Counter Preload Register (ESMLTCPR) . . . 918
17.5.16 ESM Error Key Register (ESMEKR) . . . 919
17.5.17 ESM Status Shadow Register 2 (ESMSSR2) . . . 920

Serial Communication Interface (SCI)/Local Interconnect Network (LIN) Module . . . 921
18.1 Introduction and Features . . . 922
18.1.1 SCI Features . . . 922
18.1.2 LIN Features . . . 923
18.2 Block Diagram . . . 924
18.3 SCI Communication Formats . . . 927
18.3.1 SCI Frame Formats . . . 927
18.3.2 SCI Timing Mode . . . 928
18.3.3 SCI Baud Rate . . . 928
18.3.4 SCI Multiprocessor Communication Modes . . . 931
18.3.5 SCI Multi Buffered Mode . . . 933
18.4 SCI Interrupts . . . 935
18.4.1 Transmit Interrupt . . . 936
18.4.2 Receive Interrupt . . . 936
18.4.3 WakeUp Interrupt . . . 936
18.4.4 Error Interrupts . . . 937
18.5 SCI DMA Interface . . . 938
18.6 SCI Configurations . . . 939
18.6.1 Receiving Data . . . 939
18.6.2 Transmitting Data . . . 940
18.7 SCI Low Power Mode . . . 941
18.8 LIN Communication Formats . . . 942
18.8.1 LIN Standards . . . 942
18.8.2 Message Frame . . . 942
18.8.3 Synchronizer . . . 944
18.8.4 Baud Rate . . . 944
18.8.5 Header Generation . . . 946
18.8.6 Extended Frames Handling . . . 951
18.8.7 Timeout Control . . . 952
18.8.8 TXRX Error Detector (TED) . . . 953
18.8.9 Message Filtering and Validation . . . 955
18.8.10 Receive Buffers . . . 957
18.8.11 Transmit Buffers . . . 958
18.9 LIN Interrupts . . . 959
18.10 LIN DMA Interface . . . 960
18.10.1 LIN Receive DMA Requests . . . 960
18.10.2 LIN Transmit DMA Requests . . . 960
18.11 LIN Configurations . . . 961
18.11.1 Receiving Data . . . 961
18.11.2 Transmitting Data . . . 962

18.12 Low-Power Mode 963
18.12.1 Entering Sleep Mode 963
18.12.2 Wakeup 963
18.12.3 Wakeup Timeouts 964
18.13 Emulation Mode 965
18.14 SCI/LIN Control Registers 966
18.14.1 SCI Global Control Register 0 (SCIGCR0) 971
18.14.2 SCI Global Control Register 1 (SCIGCR1) 972
18.14.3 SCI Global Control Register 2 (SCIGCR2) 979
18.14.4 SCI Set Interrupt Register (SCISETINT) 981
18.14.5 SCI Clear Interrupt Register (SCICLEARINT) 986
18.14.6 SCI Set Interrupt Level Register (SCISETINTLVL) 991
18.14.7 SCI Clear Interrupt Level Register (SCICLEARINTLVL) 995
18.14.8 SCI Flags Register (SCIFLR) 999
18.14.9 SCI Interrupt Vector Offset 0 (SCIINTVECT0) 1011
18.14.10SCI Interrupt Vector Offset 1 (SCIINTVECT1) 1012
18.14.11SCI Format Control Register (SCIFORMAT) 1013
18.14.12Baud Rate Selection Register (BRS) 1015
18.14.13SCI Data Buffers (SCIED, SCIRD, SCITD) 1018
18.14.14SCI Pin I/O Control Register 0 (SCIPIO0) 1021
18.14.15SCI Pin I/O Control Register 1 (SCIPIO1) 1022
18.14.16SCI Pin I/O Control Register 2 (SCIPIO2) 1025
18.14.17SCI Pin I/O Control Register 3 (SCIPIO3) 1026
18.14.18SCI Pin I/O Control Register 4 (SCIPIO4) 1028
18.14.19SCI Pin I/O Control Register 5 (SCIPIO5) 1030
18.14.20SCI Pin I/O Control Register 6 (SCIPIO6) 1032
18.14.21SCI Pin I/O Control Register 7 (SCIPIO7) 1034
18.14.22SCI Pin I/O Control Register 8 (SCIPIO8) 1035
18.14.23LIN Compare Register (LINCOMPARE) 1036
18.14.24LIN Receive Buffer 0 Register (LINRD0) 1038
18.14.25LIN Receive Buffer 1 Register (LINRD1) 1039
18.14.26LIN Mask Register (LINMASK) 1040
18.14.27LIN Identification Register (LINID) 1041
18.14.28LIN Transmit Buffer 0 Register(LINTD0) 1042
18.14.29LIN Transmit Buffer 1 Register (LINTD1) 1043
18.14.30Maximum Baud Rate Selection Register (MBRS) 1044
18.14.31Input/Output Error Enable (IODFTCTRL) Register 1045
18.15 GPIO Functionality 1049
18.15.1 GPIO Functionality 1049
18.15.2 Under Reset 1049
18.15.3 Out of Reset 1049
18.15.4 Open-Drain Feature Enabled on a Pin 1050
18.15.5 Summary 1050

General-Purpose Input/Output (GIO) Module 1051
19.1 Overview 1052
19.2 Functional Description of GIO Module 1053
19.2.1 GIO Block Diagram 1054
19.2.2 I/O Blocks 1054
19.2.3 External Interrupt Block 1055
19.3 Device Modes of Operation 1060
19.3.1 Emulation Mode 1060
19.3.2 Power-Down Mode (Low-Power Mode) 1060

19.4 Pullup/Pulldown Function 1061
19.4.1 Summary 1061
19.5 Open Drain Function 1062
19.6 GIO Control Registers 1063
19.6.1 GIO Global Control Register (GIOGCR0) 1066
19.6.2 GIO Interrupt Detect Register (GIOINTDET) 1067
19.6.3 GIO Interrupt Polarity Register (GIOPOL) 1069
19.6.4 GIO Interrupt Enable Registers (GIOENASET and GIOENACLR) 1071
19.6.5 GIO Interrupt Priority Registers (GIOLVLSET and GIOLVLCLR) 1074
19.6.6 GIO Interrupt Flag Register (GIOFLG) 1078
19.6.7 GIO Offset A Register (GIOOFFA) 1080
19.6.8 GIO Offset B Register (GIOOFFB) 1081
19.6.9 GIO Emulation A Register (GIOEMUA) 1082
19.6.10 GIO Emulation B Register (GIOEMUB) 1083
19.6.11 GIO Data Direction Registers [A-H][7:0] (GIODIR[A-H][7:0]) 1084
19.6.12 GIO Data Input Registers [A-H][7:0] (GIODIN[A-H][7:0]) 1085
19.6.13 GIO Data Output Registers [A-H][7:0] (GIODOUT[A-H][7:0]) 1086
19.6.14 GIO Data Set Register [A-H][7:0] (GIODSET[A-H][7:0]) 1087
19.6.15 GIO Data Clear Registers [A-H][7:0] (GIODCLR[A-H][7:0]) 1088
19.6.16 GIO Open Drain Register [A-H][7:0] (GIOPDR[A-H][7:0]) 1089
19.6.17 GIO Pull Disable Registers [A-H][7:0] (GIOPULDIS[A-H][7:0]) 1090
19.6.18 GIO Pull Select Register [A-H][7:0] (GIOPSL[A-H][7:0]) 1091
19.7 Applications 1092
19.7.1 Example: Setting Interrupts and Configuring Pins for Output 1093
19.7.2 Example: Toggling Output Buffers 1094
19.7.3 Example: Clearing Interrupt Flags and Setting Interrupts 1095
19.7.4 Example: Reading Port B Input Register 1096

. 1097

Revision History 1097

Architecture Specification

This chapter describes the architecture of the Texas Instruments Incorporated (TI) TMS470M Series of microcontrollers. It gives the internal device architecture, the memory mapping, and the main system modules.

Topic **Page**

1.1 Overview. 18
1.2 Block Diagrams 19
1.3 Memory Mapping 20
1.4 System Module (SYS) 27
1.5 Clock Definition 28
1.6 Resets. 33
1.7 Memory Ordering Model and Memory Protection 34
1.8 System Abort Concept and Illegal Transactions 36
1.9 System Interrupts 39
1.10 Memory Self Test. 40
1.11 Memory Module Hardware Initialization 42

1.1 Overview

The TMS470M Series of microcontrollers is an interconnect platform, which defines the interface between the CPUs and the modules. The platform is independent of a CPU and can be used with any ARM CPU. The system bus protocol used for the CPU interconnect varies per CPU as needed to maximize performance and minimize silicon area.

The ARM CPU is interconnected with the rest of the device via the Switched Central Resource, or SCR (see section 3.2). The SCR allows an interconnection also with other bus masters such as the slave modules, which includes the RAM (see section 5.1), the CRC module (see section 9.1), the peripheral bus bridge (see section 3.4).

All peripherals are connected to the peripheral bridge that allows the conversion from the system bus protocol to the peripheral bus protocol. The bridge also separates the system bus clock domain from the peripheral bus clock domain. The peripheral decoding and muxing are done within the Peripheral Central Resource (PCR) module (see section 3.4) that is attached to the peripheral bridge.

The SYSTEM module is accessible from the peripheral bus and contain all the register needed for the setting of the device [clock ratios, phase-locked loop (PLL) module, etc.] as well as the status flag of the system (exceptions sources, device ID, etc.) (see section 1.4).

The interrupts generated by the different sources are handled by the vector interrupt manager (M3VIM) module. The M3VIM allows backward compatibility with the TMS470R1x family and allows new vectored interrupt capability to decrease the interrupt latency (see section 8.1 for more information).

The TMS470M Series of devices incorporate a real-time interrupt (RTI) module specifically designed to support time-triggered operating systems and real-time operating systems (see section 16.1 for more information).

1.2 Block Diagrams

The following sections present general block diagrams of the TMS470M Series in commonly used configurations.

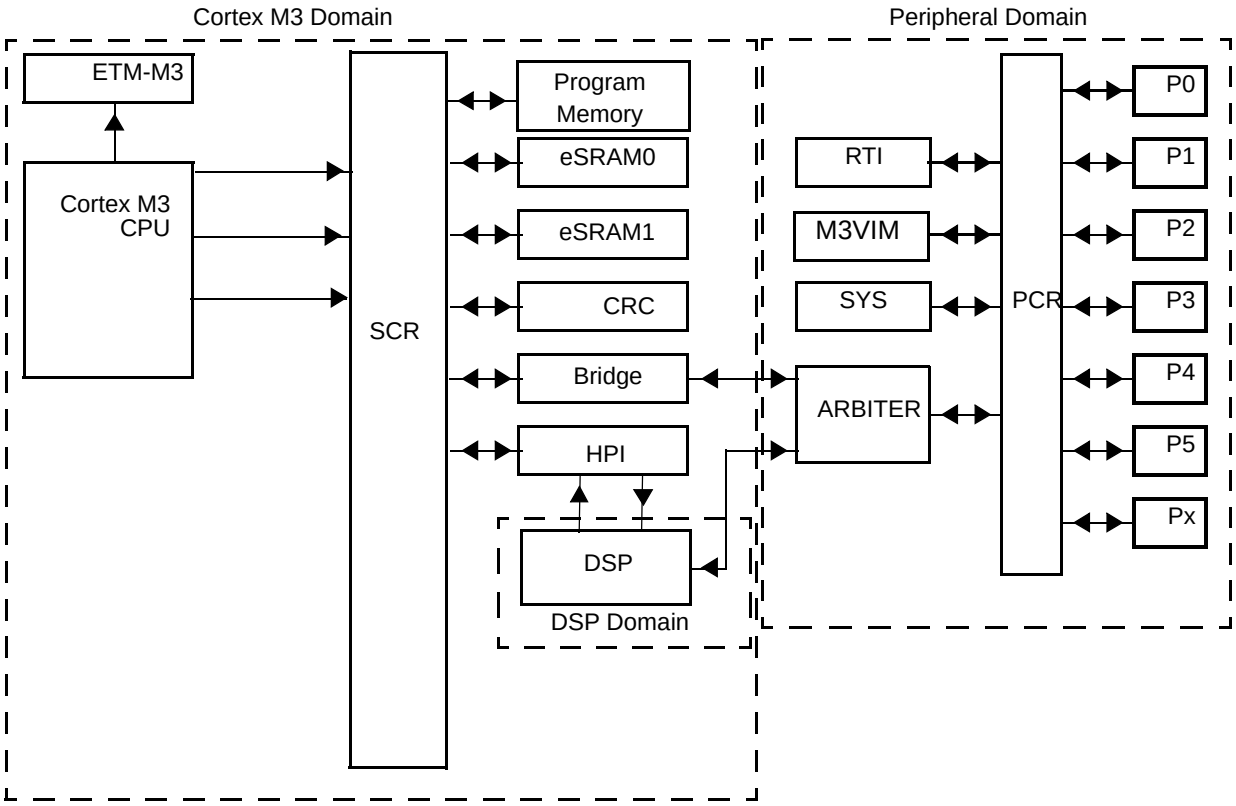
1.2.1 TMS470M Series with ARM Cortex M3 CPU

The ARM Cortex M3 CPU has three 32b extended AMBA AHB bus interfaces to memory. The ICODE bus handles instruction fetch from program memory and SRAM. The DCODE bus handles data fetch from program memory and SRAM. The SYSTEM bus handles peripheral transactions and may process instructions or data.

Figure 1-1 shows a generic TMS470M Series Microcontroller using an ARM Cortex M3 CPU.. The CPU is connected to the switched central resource. The switched central resource is a 64-bit bus matrix allowing multiple transactions per cycle. This version has the following features:

- A unified SCR allows all bus masters access to memories and peripherals.
- This configuration allows internal memory as well as an extension to external memory.

Figure 1-1. Block Diagram with ARM Cortex M3 CPU



1.3 Memory Mapping

The TMS470M Series has several options for memory mapping, described in the following sections.

1.3.1 Internal Program Memory

When using internal program memory, the memory mapping has three main blocks, which are decoded in the bus matrix decoders:

- Internal memory frame
  - eSRAM frame
  - CRC frame
  - Peripheral frame

The internal memory interface integrates a sub-decoder to provide one internal chip-select of 16 MB.

The eSRAM wrapper by construction allows a multiple eSRAM bank implementation from 8K byte up to 256K byte. The decoder allows two SRAM wrappers to be connected and a reserved space for two additional SRAM wrappers. Each eSRAM wrappers are mapped within a 64 Mbyte frame.

The peripheral bridge decodes up to 32 peripheral selects and 64 peripheral memory selects. It also decodes the address to access the system registers (Interrupt, RTI) and system memories.

A peripheral is classified either into PCS (peripheral memory chip select) or PS (peripheral select). PCS is used for peripheral memory accesses and PS is used for peripheral register accesses.

PCS and PS are subdivided into two groups: system peripherals and non-system peripherals. Accesses to system PCS are always configured in privileged mode. Accesses to system PS are specified in the peripheral user guide. For access types generated by each bus master, refer to the specification of the bus master. Figure 1-2 lists the system peripherals and their base addresses.

Note:
For clarity, Figure 1-2 does not show the quadrant of all the peripheral selects (PSs). PS 1 to PS 31 also have 4 quadrants of 256 bytes.

Note:
The memory map is device specific, based on configuration of the device. Refer to the device data sheet for available components and memory frames.

Figure 1-2. Memory Map

Frame Name	Start Address	End Address	Size	Data Access
Internal Memory CS				
nCS0	0x0000 0000	0x00FF FFFF	16 Mbytes	8/16/32/64
External Memory CS				
nCS1	0x6000 0000	0x603F FFFF	4 Mbytes	8/16/32/64
nCS2	0x6040 0000	0x607F FFFF	4 Mbytes	8/16/32/64
nCS3	0x6080 0000	0x60BF FFFF	4 Mbytes	8/16/32/64
nCS4	0x60C0 0000	0x60FF FFFF	4 Mbytes	8/16/32/64
nCS5	0x6100 0000	0x61FF FFFF	16 Mbytes	8/16/32/64
nCS6	0x6200 0000	0x62FF FFFF	16 Mbytes	8/16/32/64
nCS7	0x6300 0000	0x64FF FFFF	32 Mbytes	8/16/32/64

Figure 1-2. Memory Map (Continued)

Frame Name	Start Address	End Address	Size	Data Access
eSRAM				
CSRAM0	0x0800 0000	0x0BFF FFFF	64 Mbytes	8/16/32/64
CSRAM1	0x0C00 0000	0x0FFF FFFF	64 Mbytes	8/16/32/64
Res. CSRAM2/Boot ROM	0x1000 0000	0x13FF FFFF	64 Mbytes	8/16/32/64
Reserved CSRAM3	0x1400 0000	0x17FF FFFF	64 Mbytes	8/16/32/64
Flash SWAP location				
nCS0	0x1800 0000	0x1FFF FFFF	128 Mbytes	8/16/32/64
Flash mirror frame (Cortex R4 devices)	0x2000 0000	0x5FFF FFFF		
Reserved	0x7000 0000	0xDFFF FFFF		
Reserved ARM v7M SCS				
Private Peripheral Bus	0xE000 0000	0xE00F FFFF	1 Mbyte	8/16/32
Reserved	0xE010 0000	0xFCFF FFFF		
DSP (HPI)				
HPI frame	0xFD00 0000	0xFDFF FFFF	16 Mbytes	8/16/32
CRC				
CRC frame	0xFE00 0000	0xFEFF FFFF	16 Mbytes	8/16/32/64
Peripheral Memory CS				
PCS0	0xFF00 0000	0xFF01 FFFF	128K bytes	8/16/32
PCS1	0xFF02 0000	0xFF03 FFFF	128K bytes	8/16/32
PCS2	0xFF04 0000	0xFF05 FFFF	128K bytes	8/16/32
PCS3	0xFF06 0000	0xFF07 FFFF	128K bytes	8/16/32
PCS4	0xFF08 0000	0xFF09 FFFF	128K bytes	8/16/32
PCS5	0xFF0A 0000	0xFF0B FFFF	128K bytes	8/16/32
PCS6	0xFF0C 0000	0xFF0D FFFF	128K bytes	8/16/32
PCS7	0xFF0E 0000	0xFF0F FFFF	128K bytes	8/16/32
PCS8	0xFF10 0000	0xFF11 FFFF	128K bytes	8/16/32
PCS9	0xFF12 0000	0xFF13 FFFF	128K bytes	8/16/32
PCS10	0xFF14 0000	0xFF15 FFFF	128K bytes	8/16/32
PCS11	0xFF16 0000	0xFF17 FFFF	128K bytes	8/16/32
PCS12	0xFF18 0000	0xFF19 FFFF	128K bytes	8/16/32
PCS13	0xFF1A 0000	0xFF1B FFFF	128K bytes	8/16/32
PCS14	0xFF1C 0000	0xFF1D FFFF	128K bytes	8/16/32
PCS15	0xFF1E 0000	0xFF1F FFFF	128K bytes	8/16/32
PCS16	0xFF20 0000	0xFF21 FFFF	128K bytes	8/16/32
PCS17	0xFF22 0000	0xFF23 FFFF	128K bytes	8/16/32
PCS18	0xFF24 0000	0xFF25 FFFF	128K bytes	8/16/32
PCS19	0xFF26 0000	0xFF27 FFFF	128K bytes	8/16/32

Figure 1-2. Memory Map (Continued)

Frame Name	Start Address	End Address	Size	Data Access
PCS20	0xFF28 0000	0xFF29 FFFF	128K bytes	8/16/32
PCS21	0xFF2A 0000	0xFF2B FFFF	128K bytes	8/16/32
PCS22	0xFF2C 0000	0xFF2D FFFF	128K bytes	8/16/32
PCS23	0xFF2E 0000	0xFF2F FFFF	128K bytes	8/16/32
PCS24	0xFF30 0000	0xFF31 FFFF	128K bytes	8/16/32
PCS25	0xFF32 0000	0xFF33 FFFF	128K bytes	8/16/32
PCS26	0xFF34 0000	0xFF35 FFFF	128K bytes	8/16/32
PCS27	0xFF36 0000	0xFF37 FFFF	128K bytes	8/16/32
PCS28	0xFF38 0000	0xFF39 FFFF	128K bytes	8/16/32
PCS29	0xFF3A 0000	0xFF3B FFFF	128K bytes	8/16/32
PCS30	0xFF3C 0000	0xFF3D FFFF	128K bytes	8/16/32
PCS31	0xFF3E 0000	0xFF3F FFFF	128K bytes	8/16/32
PCS32	0xFF40 0000	0xFF41 FFFF	128K bytes	8/16/32
PCS33	0xFF42 0000	0xFF43 FFFF	128K bytes	8/16/32
PCS34	0xFF44 0000	0xFF45 FFFF	128K bytes	8/16/32
PCS35	0xFF46 0000	0xFF47 FFFF	128K bytes	8/16/32
PCS36	0xFF48 0000	0xFF49 FFFF	128K bytes	8/16/32
PCS37	0xFF4A 0000	0xFF4B FFFF	128K bytes	8/16/32
PCS38	0xFF4C 0000	0xFF4D FFFF	128K bytes	8/16/32
PCS39	0xFF4E 0000	0xFF4F FFFF	128K bytes	8/16/32
PCS40	0xFF50 0000	0xFF51 FFFF	128K bytes	8/16/32
PCS41	0xFF52 0000	0xFF53 FFFF	128K bytes	8/16/32
PCS42	0xFF54 0000	0xFF55 FFFF	128K bytes	8/16/32
PCS43	0xFF56 0000	0xFF57 FFFF	128K bytes	8/16/32
PCS44	0xFF58 0000	0xFF59 FFFF	128K bytes	8/16/32
PCS45	0xFF5A 0000	0xFF5B FFFF	128K bytes	8/16/32
PCS46	0xFF5C 0000	0xFF5D FFFF	128K bytes	8/16/32
PCS47	0xFF5E 0000	0xFF5F FFFF	128K bytes	8/16/32
PCS48	0xFF60 0000	0xFF61 FFFF	128K bytes	8/16/32
PCS49	0xFF62 0000	0xFF63 FFFF	128K bytes	8/16/32
PCS50	0xFF64 0000	0xFF65 FFFF	128K bytes	8/16/32
PCS51	0xFF66 0000	0xFF67 FFFF	128K bytes	8/16/32
PCS52	0xFF68 0000	0xFF69 FFFF	128K bytes	8/16/32
PCS53	0xFF6A 0000	0xFF6B FFFF	128K bytes	8/16/32
PCS54	0xFF6C 0000	0xFF6E FFFF	128K bytes	8/16/32
PCS55	0xFF6E 0000	0xFF6F FFFF	128K bytes	8/16/32
PCS56	0xFF70 0000	0xFF71 FFFF	128K bytes	8/16/32
PCS57	0xFF72 0000	0xFF73 FFFF	128K bytes	8/16/32
PCS58	0xFF74 0000	0xFF75 FFFF	128K bytes	8/16/32
PCS59	0xFF76 0000	0xFF77 FFFF	128K bytes	8/16/32
PCS60	0xFF78 0000	0xFF79 FFFF	128K bytes	8/16/32
PCS61	0xFF7A 0000	0xFF7B FFFF	128K bytes	8/16/32
PCS62	0xFF7C 0000	0xFF7D FFFF	128K bytes	8/16/32
PCS63	0xFF7E 0000	0xFF7F FFFF	128K bytes	8/16/32
Reserved	0xFF80 0000	0xFF9F 7FFF		

Figure 1-2. Memory Map (Continued)

Frame Name	Start Address	End Address	Size	Data Access
CoreSight Debug				
CSCS0 (Debug ROM)	0xFFA0 0000	0xFFA0 0FFF	4K bytes	8/16/32
CSCS1	0xFFA0 1000	0xFFA0 1FFF	4K bytes	8/16/32
CSCS2	0xFFA0 2000	0xFFA0 2FFF	4K bytes	8/16/32
CSCS3	0xFFA0 3000	0xFFA0 3FFF	4K bytes	8/16/32
CSCS4	0xFFA0 4000	0xFFA0 4FFF	4K bytes	8/16/32
CSCS5	0xFFA0 5000	0xFFA0 5FFF	4K bytes	8/16/32
CSCS6	0xFFA0 6000	0xFFA0 6FFF	4K bytes	8/16/32
CSCS7	0xFFA0 7000	0xFFA0 7FFF	4K bytes	8/16/32
CSCS8	0xFFA0 8000	0xFFA0 8FFF	4K bytes	8/16/32
CSCS9	0xFFA0 9000	0xFFA0 9FFF	4K bytes	8/16/32
CSCS10	0xFFA0 A000	0xFFA0 AFFF	4K bytes	8/16/32
CSCS11	0xFFA0 B000	0xFFA0 BFFF	4K bytes	8/16/32
CSCS12	0xFFA0 C000	0xFFA0 CFFF	4K bytes	8/16/32
CSCS13	0xFFA0 D000	0xFFA0 DFFF	4K bytes	8/16/32
CSCS14	0xFFA0 E000	0xFFA0 EFFF	4K bytes	8/16/32
CSCS15	0xFFA0 F000	0xFFA0 FFFF	4K bytes	8/16/32
CSCS16	0xFFA1 0000	0xFFA1 0FFF	4K bytes	8/16/32
CSCS17	0xFFA1 1000	0xFFA1 1FFF	4K bytes	8/16/32
CSCS18	0xFFA1 2000	0xFFA1 2FFF	4K bytes	8/16/32
CSCS19	0xFFA1 3000	0xFFA1 3FFF	4K bytes	8/16/32
CSCS20	0xFFA1 4000	0xFFA1 4FFF	4K bytes	8/16/32
CSCS21	0xFFA1 5000	0xFFA1 5FFF	4K bytes	8/16/32
CSCS22	0xFFA1 6000	0xFFA1 6FFF	4K bytes	8/16/32
CSCS23	0xFFA1 7000	0xFFA1 7FFF	4K bytes	8/16/32
CSCS24	0xFFA1 8000	0xFFA1 8FFF	4K bytes	8/16/32
CSCS25	0xFFA1 9000	0xFFA1 9FFF	4K bytes	8/16/32
CSCS26	0xFFA1 A000	0xFFA1 AFFF	4K bytes	8/16/32
CSCS27	0xFFA1 B000	0xFFA1 BFFF	4K bytes	8/16/32
CSCS28	0xFFA1 C000	0xFFA1 CFFF	4K bytes	8/16/32
CSCS29	0xFFA1 D000	0xFFA1 DFFF	4K bytes	8/16/32
CSCS30	0xFFA1 E000	0xFFA1 EFFF	4K bytes	8/16/32
CSCS31	0xFFA1 F000	0xFFA1 FFFF	4K bytes	8/16/32
Reserved (Future Debug)	0xFFA2 0000	0xFFAF FFFF		
Reserved	0xFFB0 0000	0xFFF7 7FFF		
Peripheral Select				
PS 31	0xFFF7 8000	0xFFF7 83FF	1K bytes	8/16/32
PS 30	0xFFF7 8400	0xFFF7 87FF	1K bytes	8/16/32
PS 29	0xFFF7 8800	0xFFF7 8BFF	1K bytes	8/16/32
PS 28	0xFFF7 8C00	0xFFF7 8FFF	1K bytes	8/16/32
PS 27	0xFFF7 9000	0xFFF7 93FF	1K bytes	8/16/32
PS 26	0xFFF7 9400	0xFFF7 97FF	1K bytes	8/16/32
PS 25	0xFFF7 9800	0xFFF7 9BFF	1K bytes	8/16/32

Figure 1-2. Memory Map (Continued)

Frame Name	Start Address	End Address	Size	Data Access
PS 24	0xFFF7 9C00	0xFFF7 9FFF	1K bytes	8/16/32
PS 23	0xFFF7 A000	0xFFF7 A3FF	1K bytes	8/16/32
PS 22	0xFFF7 A400	0xFFF7 A7FF	1K bytes	8/16/32
PS 21	0xFFF7 A800	0xFFF7 ABFF	1K bytes	8/16/32
PS 20	0xFFF7 AC00	0xFFF7 AFFF	1K bytes	8/16/32
PS 19	0xFFF7 B000	0xFFF7 B3FF	1K bytes	8/16/32
PS 18	0xFFF7 B400	0xFFF7 B7FF	1K bytes	8/16/32
PS 17	0xFFF7 B800	0xFFF7 BBFF	1K bytes	8/16/32
PS 16	0xFFF7 BC00	0xFFF7 BFFF	1K bytes	8/16/32
PS 15	0xFFF7 C000	0xFFF7 C3FF	1K bytes	8/16/32
PS 14	0xFFF7 C400	0xFFF7 C7FF	1K bytes	8/16/32
PS 13	0xFFF7 C800	0xFFF7 CBFF	1K bytes	8/16/32
PS 12	0xFFF7 CC00	0xFFF7 CFFF	1K bytes	8/16/32
PS 11	0xFFF7 D000	0xFFF7 D3FF	1K bytes	8/16/32
PS 10	0xFFF7 D400	0xFFF7 D7FF	1K bytes	8/16/32
PS 9	0xFFF7 D800	0xFFF7 DBFF	1K bytes	8/16/32
PS 8	0xFFF7 DC00	0xFFF7 DFFF	1K bytes	8/16/32
PS 7	0xFFF7 E000	0xFFF7 E3FF	1K bytes	8/16/32
PS 6	0xFFF7 E400	0xFFF7 E7FF	1K bytes	8/16/32
PS 5	0xFFF7 E800	0xFFF7 EBFF	1K bytes	8/16/32
PS 4	0xFFF7 EC00	0xFFF7 EFFF	1K bytes	8/16/32
PS 3	0xFFF7 F000	0xFFF7 F3FF	1K bytes	8/16/32
PS 2	0xFFF7 F400	0xFFF7 F7FF	1K bytes	8/16/32
PS 1	0xFFF7 F800	0xFFF7 FBFF	1K bytes	8/16/32
PS 0 Quadrant 0	0xFFF7 FC00	0xFFF7 FCFF	256 bytes	8/16/32
PS 0 Quadrant 1	0xFFF7 FD00	0xFFF7 FDFF	256 bytes	8/16/32
PS 0 Quadrant 2	0xFFF7 FE00	0xFFF7 FEFF	256 bytes	8/16/32
PS 0 Quadrant 3	0xFFF7 FF00	0xFFF7 FFFF	256 bytes	8/16/32
SYSTEM Peripherals				
Reserved	0xFFF8 0000	0xFFF8 0FFF	4K bytes	8/16/32
Reserved	0xFFF8 1000	0xFFFF 1FFF	4K bytes	8/16/32
VIM RAM	0xFFF8 2000	0xFFF8 2FFF	4K bytes	8/16/32
Reserved	0xFFF8 3000	0xFFF8 3FFF	4K bytes	8/16/32
Reserved	0xFFF8 4000	0xFFF8 4FFF	4K bytes	8/16/32
Reserved	0xFFF8 5000	0xFFF8 5FFF	4K bytes	8/16/32
Reserved	0xFFF8 6000	0xFFF8 6FFF	4K bytes	8/16/32
Flash wrapper 1	0xFFF8 7000	0xFFF8 7FFF	4K bytes	8/16/32
Flash wrapper 2 (optional)	0xFFF8 8000	0xFFF8 8FFF	4K bytes	8/16/32
ICEcrusher7E (optional)	0xFFF8 9000	0xFFF8 9FFF	4K bytes	8/16/32
Reserved	0xFFF8 A000	0xFFF8 AFFF	4K bytes	8/16/32
Reserved	0xFFF8 B000	0xFFF8 BFFF	4K bytes	8/16/32
Reserved	0xFFF8 C000	0xFFF8 CFFF	4K bytes	8/16/32
Reserved	0xFFF8 D000	0xFFF8 DFFF	4K bytes	8/16/32
Reserved	0xFFF8 E000	0xFFF8 EFFF	4K bytes	8/16/32
Reserved	0xFFF8 F000	0xFFF8 FFFF	4K bytes	8/16/32

Figure 1-2. Memory Map (Continued)

Frame Name	Start Address	End Address	Size	Data Access
Reserved	0xFFF9 0000	0xFFFE FFFF		
Reserved	0xFFFF 0000	0xFFFF 7FFF	32K bytes	8/16/32
Reserved	0xFFFF 8000	0xFFFF DFFF		
PCR Registers	0xFFFF E000	0xFFFF E0FF	256 bytes	8/16/32
System Frame 2 Registers	0xFFFF E100	0xFFFF E1FF	256 bytes	8/16/32
RAM ECC regs 3	0xFFFF E200	0xFFFF E2FF	256 bytes	8/16/32
RAM ECC regs 4	0xFFFF E300	0xFFFF E3FF	256 bytes	8/16/32
Memory BIST controller 1	0xFFFF E400	0xFFFF E4FF	256 bytes	8/16/32
Memory BIST controller 2	0xFFFF E500	0xFFFF E5FF	256 bytes	8/16/32
Logic BIST controller 1	0xFFFF E600	0xFFFF E6FF	256 bytes	8/16/32
Logic BIST controller 2	0xFFFF E700	0xFFFF E7FF	256 bytes	8/16/32
EMIF 1 registers (opt.)	0xFFFF E800	0xFFFF E8FF	256 bytes	8/16/32
EMIF 2 registers (opt.)	0xFFFF E900	0xFFFF E9FF	256 bytes	8/16/32
Reserved	0xFFFF EA00	0xFFFF EAFF	256 bytes	8/16/32
Reserved	0xFFFF EB00	0xFFFF EBFF	256 bytes	8/16/32
Reserved	0xFFFF EC00	0xFFFF ECFF	256 bytes	8/16/32
Wakeup registers	0xFFFF ED00	0xFFFF EDFF	256 bytes	8/16/32
RTI2 registers (opt.)	0xFFFF EE00	0xFFFF EEFF	256 bytes	8/16/32
Pin mux control (opt.)	0xFFFF EF00	0xFFFF EFFF	256 bytes	8/16/32
Reserved	0xFFFF F000	0xFFFF F3FF	1K bytes	8/16/32
Reserved	0xFFFF F400	0xFFFF F4FF	256 bytes	8/16/32
ESM registers	0xFFFF F500	0xFFFF F5FF	256 bytes	8/16/32
CPU Compare Module (opt.)	0xFFFF F600	0xFFFF F6FF	256 bytes	8/16/32
DMM registers (optional)	0xFFFF F700	0xFFFF F7FF	256 bytes	8/16/32
RAM ECC regs 2 (opt.)	0xFFFF F800	0xFFFF F8FF	256 bytes	8/16/32
RAM ECC regs 1 (opt.)	0xFFFF F900	0xFFFF F9FF	256 bytes	8/16/32
Reserved	0xFFFF FA00	0xFFFF FAFF	256 bytes	8/16/32
Reserved	0xFFFF FB00	0xFFFF FBFF	256 bytes	8/16/32
RTI	0xFFFF FC00	0xFFFF FCFF	256 bytes	8/16/32
VIM Parity register	0xFFFF FD00	0xFFFF FDFF	256 bytes	8/16/32
M3VIM registers	0xFFFF FE00	0xFFFF FEFF	256 bytes	8/16/32
System module registers	0xFFFF FF00	0xFFFF FFFF	256 bytes	8/16/32

1.3.2 Endian System

The TMS470M Series uses a word invariant big endian programmer's model; the rules in Table 1-1 apply.

Table 1-1. Active Byte Lanes for a 32-Bit Data Bus

Transfer size	Offset Address	DATA 31:24	DATA 23:16	DATA 15:8	DATA 7:0
Word	0	✓	✓	✓	✓
Half-Word	0	✓	✓		
Half-Word	2			✓	✓
byte	0	✓			
byte	1		✓		
byte	2			✓	
byte	3				✓

1.4 System Module (SYS)

The system module has several functions.

- It controls device operations such as:
  - Exception generation
  - Clock control functions
  - Device identification
  - Global configuration registers
  - Software interrupt registers (4 interrupts)
  - Reset generation
- It logs exception events in the exception status registers.
- It contains all the configuration registers for the following components:
  - Global clock module and PLL module
  - Bus matrix module (priority programming and memory swapping)
  - Peripheral bridge (bus error logging and clock ratio)
  - eSRAM wait-state programming
  - External clock prescaler (ECP) module
  - Voltage regulator (on a specific device only)

1.5 Clock Definition

The following sections describe an overview of the TMS470M Series clocks. For more detailed information, please refer to the Global Clock Module (GCM) specification.

1.5.1 Terminology

A *primary clock source* is a source that generates a clock independent of other circuitry. Typical examples of primary clock sources are internal oscillators that drive external crystal/resonators, paths for driven external clock, or internal free-running ring oscillators.

A *secondary clock source* is a source that generates a clock based on the primary clock source. Examples of secondary clock sources are PLL and FLL outputs.

An *initiator* is a module that can apply a request onto a bus.

A *target* is the recipient of a request on a bus.

A *central resource* defines a bus interconnect.

A *bridge* defines a bus domain separation.

1.5.2 Clock Domains

All TMS470M Series device clocks are generated by the global clock module (GCM). The device is divided into eight different clock domains:

- CPU clock domain
- System bus clock domain
- System peripheral clock domain
- Peripheral clock domain
- Primary asynchronous peripheral clock domain
- Secondary asynchronous peripheral clock domain
- Primary real-time interrupt clock domain
- Secondary real-time interrupt clock domain

1.5.2.1 Description of Clock Domains

Table 1-2 compares the various clock domains and their specifications.

Table 1-2. Clock Domain Descriptions

Domain	Clocked By	Generated By	Comments
CPU Clock Domain	GCLK	Any clock source. By default, GCLK is clocked by OSCIN.	The GCLK frequency is determined by PLL control registers, PLLCTLx. GCLK controls all the CPU subsystems, including the caches and the MPU. MCLK is derived from GCLK. Note: On some cores, such as ARM7, the core clock is referenced as MCLK, as named by the core clock input port.
System Bus Clock Domain	HCLK	Always the same clock source as GCLK, although HCLK could be active while GCLK is held static.	The ratio between GCLK and HCLK is always 1.
System Peripheral Clock Domain	VCLK_sys	Derived from HCLK	VCLK_sys is the primary clock for the VBUS clock domain. It is used for all the bus transactions as well as the system peripherals (system module, M3VIM, etc.), with the exception of the real-time interrupt (RTI) module. The ratio between HCLK and VCLK_sys is programmable from 1 to 16 by the VCLKR field in the clock control register (see section 4.2.28), except for specific devices. Refer to the device-specific data sheet for more information. The default ratio value of the VCLKR is 1:2. HCLK and VCLK_sys share the same enable bit.
Peripheral Clock Domain	VCLK_periph and VCLK2	Derived from HCLK	VCLK_periph is the primary clock for the peripherals. VCLK_periph is synchronous with VCLK_sys (ratio 1:1), but it can be shut down separately. VCLK2 is a second VBUS clock domain that is used by some peripherals to run faster than VCLK_periph (see the peripheral module specifications and device-specific data sheet). The relationship between VCLK2, HCLK, and VCLK is as follows: HCLK $\geq$ VCLK2 $\geq$ VCLK_sys The ratio between HCLK and VCLK2 is programmable from 1 to 16; the default ratio value is 1:2. The GCM only supports integer ratios between VCLK_sys, VCLK2, and HCLK.
Primary and Secondary Asynchronous Clock Domains	Programmed in the system module registers (see section 4.2.20)	Any clock source	The asynchronous clock domains are used specifically for communication modules that do not support frequency-modulated clocks. The two clock domains could be used if different communication modules are used (CAN, TTCAN, Flexray, etc.). These clock domains use different PLLs than the rest of the system, therefore creating an asynchronous clock domain.
Primary and Secondary Real-Time Interrupt Clock Domains	RTICK1	Oscillator or VCLK. Other low jitter sources could be used.	By default, RTICLK1 uses the same ratio as VCLK_sys. The RTICLK1 switches to OSCIN under the following conditions: The RTI timebase is set to OSCIN in the software. The device enters stand-by mode. The PLL is shut down. RTICLK2 (the secondary RTI clock domain) is only available if a second RTI or a real-time clock (RTC) is present. RTICLK2 can also be connected to an OSCIN.

1.5.3 Clock Domains and Low Power Mode

The TMS470M Series clock domains can be disabled as described in the following sections to provide a better granularity in power consumption. The ability to disable the domains also provides a choice for secondary source during low-power mode.

1.5.3.1 Low Power Mode Hierarchy

All modules use a defined clock suspend/wakeup interface. That is, for each clock used by a target, the target has a clock stop request input and acknowledge outputs. When all targets on a specific clock domain have acknowledged a clock stop request, then the entire clock domain is disabled.

Note:
When an initiator accesses a target whose clock is stopped, the clock is restarted *for the accessed target* without awakening all other targets in that clock domain. After the access is complete, a clock stop request is re-asserted, and the target re-enters low power mode.

Wakeup should occur by interrupt/wakeup circuitry or by the CPU clearing the clock domain disable bit, rather than by an initiator access to a target whose clock is stopped.

1. When all dependent clock domains are disabled, secondary sources may be disabled.
2. When all dependent clock domains and dependent secondary sources are disabled, primary sources may be disabled.

The clock domains and sources are awakened in the reverse order.

1. The primary source is awakened.
2. The dependent secondary sources are awakened.
3. The dependent clock domains are awakened.
4. The entire clock domain is awakened.

1.5.3.2 Clock Tree Low Power Mode

Setting the clock domain disable bit for a specific clock domain (see section 4.2.16–section 4.2.18) activates a clock stop request signal for all targets on the clock domain. When all targets' acknowledgements are received, then the clock domain is disabled at the source. Each clock domain has an independent bit (except for the systems peripheral, which shares its clock domain. This clock domain is disabled with HCLK).

1.5.3.3 Clock Selection

When the CPU enters/exits low power mode, it is possible to automatically change the clock source for HCLK, VCLK, and VCLK2 domains. This flexibility is provided so that the user can choose which clocks are active while the CPU is inactive versus which clock is active on CPU wakeup (power versus performance trade-off) (see section 4.2.19).

Similarly, other clock selection muxes allow the user to set the source for the respective clock domains. Every clock selection mux change sources in a glitch-free manner (see section 4.2.19 to section 4.2.21).

All muxed clock domains (A_VCLK, RTICLK,...) default to VCLK_sys as source.

1.5.4 Primary/Secondary Clock Source Control

A clock source cannot be disabled while it is being used by an active clock domain (see section 4.2.21). In the case of a primary clock source, the clock must not be used by any clock domain or secondary source, whether active or not. After this condition is met, the clock may be disabled. After being disabled, if the clock is required, then it must be re-enabled (see section 4.2.13–section 4.2.15.).

Note:
It is not possible to switch a clock domain to a nonexistent clock source or to a clock source that is not ready. In either case, any write to the clock source is discarded.

1.5.5 Oscillator Fail Detection

The oscillator fail detector is a gross failure detection mechanism intended to detect an open circuit or a short circuit on a crystal pin. It determines whether the external reference input—either the crystal or the oscillator—is stuck low, stuck high, or running at a frequency too slow as determined by an internal reference oscillator. Depending on the type of PLL and the clock monitor circuitry implementation, an external reference source running too fast can be detected as an oscillator failure.

The oscillator fail functionality is implemented as a part of the clock module. The system module has a flag that indicates an oscillator fail flag in the GLBSTAT register (see section 4.2.54) and the reset on oscillator flag in the SYSESR register.

1.5.6 ECP

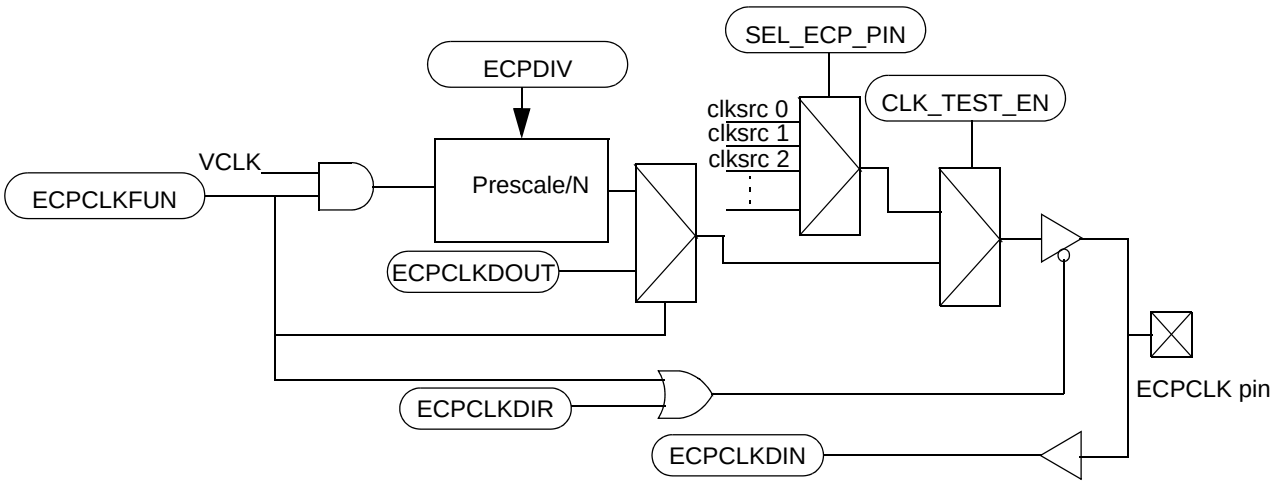
The ECP allows the device to output a continuous external clock on an I/O pin. The external clock (ECLK) frequency is a user-programmable ratio of the interface clock (VCLK_sys) frequency.

1.5.6.1 ECP Block Diagram

The ECP block diagram (see Figure 1-3) shows the overall functionality of the ECP module. The VBUS clock (VCLK) input is divided by the user-selected value programmed into the ECPCNTL register (see section 4.2.49). The output of the divider is then multiplexed with the selected ECPCLK_DOUT register. The output of the multiplexer is controlled by the ECP functional register bit (ECPCLKFUN), and is output on the external I/O pin of the device.

For Test purpose, the ECP allows the internal clock sources to be output on the ECLKPCLK Pin. By enabling the CLK_TEST_EN bit field in the clock test register (see section 4.2.35). The clock source is selected by using the SEL_ECP_PIN bit field in the same register.

Figure 1-3. ECP Block Diagram



The ECPDIV provides a programmable prescaler for generating ECLK from VCLK.

$$ECPCLK = \frac{VCLK}{(ECPDIV + 1)}$$

1.5.6.2 ECP Prescaler

The ECP prescaler generates the ECPCLK signal by dividing VCLK to a lower frequency; the prescaler is implemented as a digital counter programmable via the ECPCNTL register (ECPDIV[15:0]). The prescaler counts VCLK edges, and when the counter reaches the value programmed in ECPDIV[15:0], an ECPCLK edge is generated.

The prescaler can be programmed with divider values ranging from a minimum of 1 to a maximum of 65536. The divider values are programmed into the ECPDIV[15:0] bits in the ECPCNTL register (ECPCNTL[15:0]).

When the ECPDIV[15:0] bits are 0x07, the prescaler is set to divide by 8. The default state after reset is divide by 2.

Note:
The duty cycles of the ECP clock is 50% by design. However, depending on the I/O used and the load on the pad, some deviation from the design percentage might be observed.

The prescaler divider may be changed while the ECP module is enabled. However, to assure a smooth transition between output frequencies, the new divider value does not take effect until the current ECLK period is complete.

1.5.6.3 ECP Enable

The ECPCLKFUN bit controls the external I/O pin function of the device, as shown in Table 1-3.

Table 1-3. Settings for ECPCLKFUN

ECPCLKFUN	ECPCLKDIR	External I/O Pin Function
0	0	GIO input
0	1	GIO output
1	x	ECLK output

The multiplexer output is controlled via the ECPCLKFUN bit in the SYSPC1 register. If the ECPCLKFUN bit is set, the multiplexer outputs ECPCLK on the external I/O pin of the device. If ECPCLKFUN is cleared, the multiplexer allows control of the external pin by the GIO control registers (SYSPC2 to SYSPC9 register)

1.5.6.4 Power Down

When the device begins powering down, the VCLK_sys input is disabled. Therefore, the ECLK output is also disabled. After waking up from low-power mode, the ECP returns to its previous state, removing the need to reconfigure the ECPCTRL register after the ECP wakes up from VCLK_sys off.

1.5.6.5 Suspend

When the ECP is in suspend mode, the ECP control register can still be modified. This allows the user to modify the control register when a JTAG emulator/debugger tool is being used.

When the ECPCOS bit (ECPCNTL.23) is set, the ECP continues to output its ECPCLK if the ECPCLKFUN bit is set when the ECP is in suspend mode.

1.6 Resets

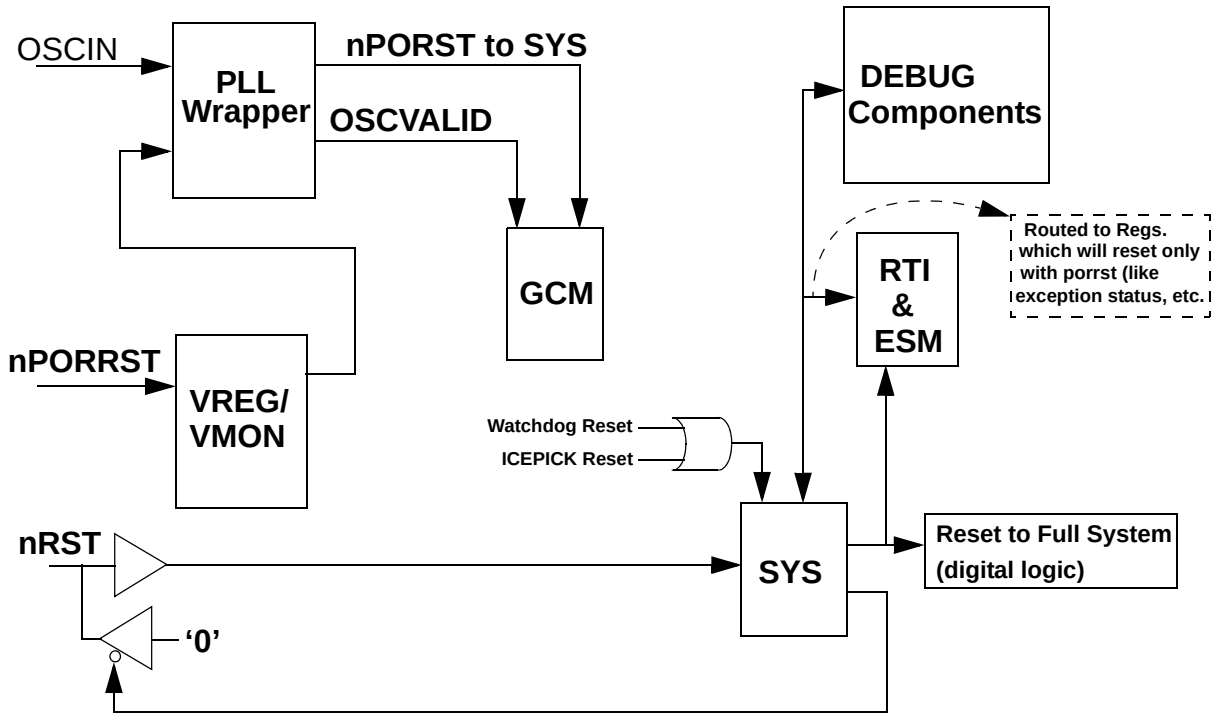
A TMS470M Series device reset can be caused by any of the conditions listed in Table 1-4.

Table 1-4. Causes of TMS470M Series Device Resets

Condition	Description
External cold nPORST pin	This is an active LOW power up reset signal.
External warm RESET pin	This is an active LOW warm reset generated by TMS470M Series system or by another external device
Watchdog	A watchdog (WD) reset occurs when either the WD timer times out, or a wrong key is written to the register.
Oscillator failure	This determines whether the external reference input, either the crystal or the oscillator, is stuck-low, stuck-high, or running at a frequency too slow as determined by an internal reference oscillator
Software reset	Writing an incorrect pattern to the software reset bits causes a system to reset.
Voltage regulator	For devices with an on-chip voltage regulator, a reset is generated when the voltage regulator detects an over or under voltage scenario relative to its internal reference voltage.
Debug reset	This signal is asserted by icepick logic and is active LOW reset. Additional information is available in the ICEPICK section.

Figure 1-4 shows how all the resets are connected within a device.

Figure 1-4. Reset Tree Diagram



nRST & nPORST from PLLW will cause SYS_nRST (async).

All the different reset conditions are flagged within the system exception register.

The reset logic holds the device in a reset state for at least eight VCLK cycles. During a reset, while the reset logic holds the device in the reset state, the external RESET signal is driven LOW. This change in the signal occurs whether the reset is caused by an internal exception or an external exception. The time the external

reset should be active to be recognized is dependent on a deliberate recognition delay. This delay before recognition avoids any unnecessary resets caused by application glitches. For specific information on glitch filter implementation, please reference the device specific datasheet for details.

1.7 Memory Ordering Model and Memory Protection

A memory ordering model describes the interaction of a bus master to memory. The CPUs used in the TMS470M Series platform architecture are compliant to the ARM Protected Memory System Architecture (PMSA) or ARM Virtual Memory System Architecture (VMSA). Full documentation of these models may be found in the *ARM Architecture Reference Manual* (ARM lit number DDI0308). The following section describes PMSA/VMSA compliant memory ordering model used in the TMS470M Series platform architecture.

1.7.1 Memory Attributes

Memory attributes describe the properties of a memory.

- Bufferability describes whether write accesses to a memory may be buffered.
- Cacheability determines whether a memory may be cached. Further sub-attributes may describe the cache policy (write-back, write-through, etc.)
- Shareability determines whether a memory is shared between multiple bus masters.

1.7.2 Memory Types

The most commonly used combinations of memory attributes are grouped together to form memory types. The three memory types defined are:

- Strongly Ordered (SO) memory, such as system critical control registers
- Device memory, such as standard memory mapped peripheral registers
- Normal memory, such as standard instruction or data memory

Table 1-5 describes the memory types.

Table 1-5. Memory Types

Memory Type	Sharable	Other	Description
Strongly Ordered	Always Shared		SO transactions always occur in program order. SO accesses are never cachable, never bufferable, and always shared.
Device	Shared	Bufferable or non-bufferable	Memory mapped peripherals shared by multiple processors
	Non-shared	Bufferable or non-bufferable	Memory mapped peripherals used by a single processor
Normal	Shared	Non-cachable, WT cachable, or WBWA cachable; bufferable or non-bufferable	Memory shared between multiple processors
	Non-shared	Non-cachable, WT cachable, or WBWA cachable; bufferable or non-bufferable	Memory used by a single processor

1.7.3 Memory Ordering Model Variations

The ARM memory ordering model has gone through many changes as the ARM architecture has evolved. As such, there are minor differences in the implementation between the different cores used in the TMS470M Series platform architecture.

1.7.3.1 ARM Cortex M3

The ARM Cortex M3 processor is compliant to PMSAv7M. The core implements a fixed background memory map with the following attributes. The background memory map can be overridden with memory regions defined by the MPU.

The Cortex M3 processor is designed with an integrated MPU. This is an optional component per ARM but is mandatory for use on Cortex M3 designs used in the TMS470M Series automotive platform. The MPU is programmed via memory mapped registers in the System Control Space. The user has 8 programmable regions available, each of which may be split into 8 sub-regions. The user has full control of memory access permissions, memory type, memory attributes, and region size.

1.8 System Abort Concept and Illegal Transactions

The system abort concept defines the response of bus masters to errors in the system. In the TMS470M Series platform architecture the system abort and error handling concept incorporates the following error types:

- Access to an illegal address (a non-implemented address)
- Access to a protected address (protection violation)
- Error on a valid access (parity, ECC, or timeout)

1.8.1 Abort Types

Two general types of aborts are possible in the system concept: precise (synchronous) and imprecise (asynchronous) aborts.

A precise abort is defined as an abort in which the initiator of a transaction can determine the exact transaction which generated the abort. The initiator may then “rewind” the system back to the pre-abort state. Because the transaction which generated the abort is known, data from this transaction such as the address, read vs. write, etc. may be recorded for later debug. A fault on a CPU instruction fetch is an example of a precise abort.

An imprecise abort is defined as an abort in which the initiator of a transaction cannot determine the exact transaction which has generated the abort. The initiator has no guarantee that the system state before the abort may be restored. A fault on a buffered write transaction is an example of an imprecise abort.

An imprecise abort may be defined as internal or external. An internal imprecise abort occurs due to buffering or other imprecision inside a bus master. Such imprecision is generally comprehended by the bus master. An external imprecise abort occurs due to buffering or other imprecision in the data path of a bus master, but outside the bus master. The bus master cannot directly comprehend such an abort and the system impact is often fatal.

The TMS470M Series platform architecture applies techniques at the system level to mitigate the impact of external imprecise aborts. System level adoption of write status sidebands to the datapath allow bus masters to comprehend external 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 external imprecise abort may be registered at the point of imprecision for later analysis.

1.8.2 Accesses to Illegal Addresses

Accesses to non-implemented addresses for the slaves on the AMBA bus and the peripherals on the VBUS is defined as an illegal address. The definition of a non-implemented address is described in Table 1-6.

Table 1-6. Definition of a Non-Implemented Address

Slave Memory Frame	Definition of Non-Implemented Address (Illegal Address)
EMIF	Accesses to deactivated EMIF chip selects are defined as accesses to a non-implemented address. The EMIF generates an error response for any access to deactivated chip selects.
eSRAMx	Accesses to non-implemented RAM banks in the eSRAMx frame are defined as accesses to a non-implemented address. The eSRAMx wrapper generates an error response for any access to the non-implemented RAM banks.
Reserved RAM frame	If the reserved RAM frame (nCSRAM2 and nCSRAM3) is not implemented, the frame is defined as a non-implemented address. In this case, the SCR generates an error response on any access to this frame. If the reserved RAM frame is implemented, the eSRAMx wrapper generates an error response for any access to any non-implemented addresses in the RAM frame.
Reserved memory	The SCR generates an error response for any access to the reserved memory.
HPI slave	If the reserved HPI slave frame is not implemented, the frame is defined as a non-implemented address. In this case, the SCR generates an error response on any access to this frame. If the HPI slave frame is implemented, the HPI generates an error response for any access to any non-implemented addresses.

Slave Memory Frame	Definition of Non-Implemented Address (Illegal Address)
CRC slave	The CRC module generates an error response for any access to the non-implemented addresses in the CRC slave frame.
Peripheral memory chip select frame	Accesses to non-implemented peripheral memory selects generate error responses. If a peripheral memory chip select is mapped to an peripheral memory that is not fully implemented, an error response is generated starting from the address to the nearest power of 2 with a minimum of 1K bytes granularity.
Peripheral select frame	Each peripheral select frame contains 4 quadrants of 256 bytes each. An error response is generated for any access to the non-implemented peripheral select quadrants.
Within the slave memory frame	An error response is generated when the AMBA/VBUS address bus accesses a non-implemented area in the slave. Every slave generates a response based on the type of transaction.

1.8.3 Illegal Accesses

Accesses to protected memory on the AMBA bus and the peripherals register or memory frames on the VBUS are defined as illegal accesses. The following discusses the different types of illegal accesses and how they are handled.

- Memory access protection.
  Memory access permission for the memory on the AMBA bus is provided by the MMU/MPU through permission faults. The 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.
- Peripheral register/peripheral memory access right violation.
  The device peripherals accesses can be protected either through the MMU/MPU by programming the level descriptors to generate permission faults or through the peripheral register frame/peripheral register frame protection register in the system module.
  - *Peripheral Register Access Violation:*
    The PCR module registers PPROTSETx contain one protection bit per peripheral register quadrant . The protection bits define the access rights to the peripherals on the VBUS. When the CPU attempts to write to any peripheral register frame for which it does not have the correct permission rights, a protection violation is detected and an error response is generated. In addition, peripherals on the VBUS may define certain register bits that are not writable in certain operating modes (refer to module register specifications to see the bits marked as RWP— read in all modes, write in privilege mode only). When the CPU attempts to write to these peripheral register bits for which it does not have the correct permission rights, the write operation is simply ignored and no memory access violation is detected.
  - *Peripheral Memory Access Violation:*
    The PCR module registers PMPROTSETx have one bit per peripheral memory frame. The protection bits define the access rights to the peripheral memory frames on the VBUS. When the CPU attempts to write to any peripheral memory frame for which it does not have the correct permission rights, a protection right access is detected and an error response is generated.

1.8.4 Illegal Transaction Detection and Response

Almost all illegal transactions provide an error response to the initiator **except** in the following cases:

- On an error response (illegal transaction) to non-cacheable buffered stores, cacheable reads, writes, swaps (linefills), some ARM CPUs do not guarantee to recover the precise instruction that caused that transaction. This loss of instruction is defined as an *imprecise abort*. For these imprecise transactions on the AMBA instruction and data buses, the SCR provides the system module with the address for an imprecise transaction.

Note:
For further explanation of imprecise aborts, please refer to the *ARM Architecture Reference Manual* (ARM reference number DDI0100). Please also refer to the CPU documentation to verify the CPU behavior is consistent with a case of imprecise abort.

- Because of write buffering in the VBUS bridge for illegal transactions on writes to the VBUS peripherals, the VBUS bridge gives an OKAY response to the initiator before the write transaction error is detected. The VBUS bridge provides the system module the address, system mode (user/privileged), and the initiator ID of the illegal transaction.

Table 1-7 is a summary of the TMS470M Series system responses when an illegal transaction is detected.

Table 1-7. System Responses to Illegal Transactions

Access Type	System mode	Illegal Transaction Response
Error response to precise CPU transactions		
Non-cacheable, non-bufferable CPU accesses (NCNB)	User/Privileged	The system generates an external abort for an error response from the slave. The virtual address of the access and the abort status are logged in the MPU fault address register (FAR) and the MPU fault status register (FSR), respectively.
Non-cacheable bufferable CPU reads (NCB)	User/Privileged	For an error response from the slave, the system generates an external abort. The virtual address of the access and the abort status are logged in the MPU fault address register (FAR) and the MPU fault status register (FSR) respectively.
Error response to imprecise CPU transactions		
Non-cacheable, bufferable writes (NCB)	All modes	The address of an illegal NCB is logged in the IMPFTADD register in the system module register (see section 4.2.39). The status is given by the IMPFASTS register (see section 4.2.38)
	User/Privileged	The system generates an imprecise abort. The CPU enters the abort routine.
TCM writes	All modes	The auxiliary FSR indicates which TCM bank caused the error. This information is overwritten by subsequent TCM write errors. See *ARM CPU Technical Reference Manual*.
	User/Privileged	The CPU enters the abort routine.
Illegal transactions on which the initiator receives an okay response		
VBUS writes	All modes	The address of an illegal transaction on the VBUS is logged in the IMPFTADD register in the system module register (see section 4.2.49). The status is given by the IMPFASTS register (see section 4.2.38)
	User/Privileged	The system generates an imprecise abort. The CPU enters the abort routine.

1.9 System Interrupts

The system generates the system software interrupts.

1.9.1 System Software Interrupt (SSI)

A system software interrupt (SSI) is generated by writing the correct key value to either of SSIRx (x=1,2,3,4) registers. While writing the key in the SSIRx register, the user can write an 8-bit value in the register. This value will then be provided to the ISR when reading the SSIVEC register. The SSIRx share the same interrupt request line (see section 4.2.57).

The source of the system software interrupt is available in the system software interrupt vector register (SSIVEC, see section 4.2.56).

1.10 Memory Self Test

The TMS470M Series control logic provides the capability to perform a built-in self test on all the RAMs in the TMS470M Series devices.The memory self-test control logic is based on the concept of implementation of hardware memory BIST (built in self-test). Multiple hardware memory BIST solutions are available for use in TMS470M Series devices. The two most commonly used memory BIST solutions are PBIST and MBIST. Additional memory self test coverage can be applied via software driven CPU based test patterns. Such a memory test solution is known as CPU memory BIST.

1.10.1 MBIST

MBIST is an industry standard memory BIST solution. MBIST logic implements fixed function memory test logic. Algorithmic test changes are not possible without a complete resynthesis of the MBIST logic. Given this constraint, MBIST is often supplemented with CPU memory BIST. MBIST is implemented as single or multiple distributed test controllers.

1.10.2 Memory BIST Implementation

Regardless of the memory BIST solution implemented, the TMS470M Series implements memory BIST for all RAMs.

The optimal memory BIST configuration varies based on device memory configuration and application. Please refer to the device specific datasheet and DFT specification to determine the exact configuration supported by your hardware.

1.10.3 Memory BIST Algorithms

The algorithms applied to the memories are a combination of industry standard tests such as the March 13N and TI proprietary algorithms. 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 algorithms are selected through the MSTGCR register (section 4.2.23, *Memory Self-Test Global Control Register (MSTGCR)*, page 96).

1.10.3.1 Checkerboard and Inverse Checkerboard

Simple march type algorithm, using checker board data pattern. Checker board data value takes into account the topology of the memory. Targets bridge coupling faults (BF) which occur when a short, or bridge, exists between two or more cells or signals.

1.10.3.2 March13N, background 0:

March13N run with background 0.

- 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

1.10.3.3 March11N, background A:

Reduced version of March13N (two read operations removed) with background 0xA

1.10.3.4 March13N, background 3, 0F, and 69:

March13N run 3 times with different background patterns 0x3, 0x0F, and 0x69

1.10.3.5 PMOS Address Decoder Algorithm:

Standard Address Decoder Algorithm targeting address decoder open faults.

The algorithm is based on writing a value to a test address and checking if the base address value changes.

The address decoder fault is associated with a pair of addresses: the base address, and the testing address. Therefore address decoder faults are always detected for a pair of memory addresses.

To detect open faults in an address decoder, the algorithm writes to a neighboring address with a hamming distance of one, and checks if this write also writes into the base cell.

1.11 Memory Module Hardware Initialization

The TMS470M Series system provides the capability to perform an hardware initialization on all modules with 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 parity (odd/even) or ECC. An example would be after power-on reset, the content of the memory arrays would be unknown and on triggering hardware initialization counter it would program the memory arrays to the corresponding values based on the error checking scheme. Another example would be triggering off the hardware initialization logic after a memory self-test run, because the arrays will be programmed to all ones after the run. Device specification will indicate which memories have this capability.

1.11.1 Memory Module Hardware Initialization Features

- Supports up to 32 memory modules on the system bus and the peripheral bus.
- Can run all the memory module initialization in parallel
- Can run each memory module initialization individually
- Captures memory module initialization completion results in status registers.

The hardware memory initialization protocol is as follows:

1. Initiate the global hardware memory initialization key (MIINTIGENA) by programming a value of 1010 to bits 3:0 of the MINITGCR.
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 MSIENA register.
3. If the global hardware is enabled, on detecting a write of a value 1 to the MSIENAn bit of the MSIENA register a pulse is generated on the corresponding MMISTART(31–0) signal, which triggers a hardware initialization on the corresponding module it is mapped to. Which module it is mapped to is device specific and will be provided in the device-specific data sheet.
4. On detection of a pulse on its MMISTART (memory module initialization) register, the corresponding module will initialize its memories based on its memory error checking scheme (even parity or odd parity or ECC).
5. When the memory initialization is complete, the module will generate a pulse on its MMIDONE (memory module initialization done) signal, which sets the corresponding bit in the system module MINISTAT register to indicate the completion of its memory initialization.
6. When all the done status bits corresponding to the selected modules are set, then the memory hardware initialization done bit (MINIDONE) is set in the MSINIGSTAT register.

Low Power Mode Operation

This chapter describes the low power mode architecture of the Texas Instruments Incorporated (TI) TMS470M Series of microcontrollers.

Topic **Page**

2.1 Overview 44
2.2 Standard Power Management Functionality 44

2.1 Overview

The TMS470M Series platform architecture is intended for use in a variety of applications. The power requirements for these applications may vary greatly. As such, a highly flexible system of power management is provided so that the system designer may trade off power versus functionality as needed in their application.

Devices with standard power management capabilities utilize the SYSTEM module controls to enable basic power management via clock and voltage regulator controls. Standard power management capabilities are intended for use in single processor systems without complex autonomous low power peripherals.

2.2 Standard Power Management Functionality

TMS470M Series devices support multiple standard low power modes. These different modes allow the user to trade-off the amount of current used during low power mode versus functionality and wakeup time.

This document is intended to describe all the possible low power modes supported by the TMS470M Series architecture. Which actual modes are supported on a specific TMS470M Series device is documented in the device-specific datasheet.

2.2.1 Overview

The following sections discuss the different low power modes supported by TMS470M Series devices. Modes are defined in order from highest current to lowest current.

All TMS470M Series low power modes have the following common characteristics and these conditions must be met in order for the device to be considered to be in a low power mode.

- The CPU and system clocks are disabled.
- The PLL is disabled.
- Flash banks and pump are in sleep mode.
- All peripheral modules are in low power modes and clocks are disabled. (Exceptions to this may occur and would be documented in the TMS470M Series device-specific datasheet).

Because the TMS470M Series architecture is defined with ultimate flexibility in enabling and disabling clocks, there are more possible modes than those defined in this document; however, these are the only modes actually characterized or specified for TMS470M Series devices. Please consult the specific TMS470M Series device datasheet for the modes supported by the device.

2.2.2 Active Clock Modes

An *active clock* mode is defined as any low power mode that has at least one clock still running. This mode allows the user to wake up the device more quickly than low power modes with no clocks running. Active clock modes allow the device to wake up on an internal source without external support. All active clock modes support the following wakeup (return-to-normal operation) methods:

- Any wakeup interrupt from a currently clocked peripheral:
  - Interrupt or wakeup signal from the RTI (RTI clock must still be enabled)
  - Any external GIO interrupt defined as a wakeup interrupt
  - CAN message
  - SCI or LIN message
- Power-on or system reset

2.2.2.1 Doze Mode

Doze mode is defined as follows:

- The primary oscillator is enabled.
- The RTI clock can be either enabled or disabled (depends on user setting).
- The low-frequency clock (either the 32 KHz oscillator or the low-frequency low power oscillator (LF LPO)) source can be either enabled or disabled by the user.
- The voltage regulator (VREG), if present, must be put in LPM1 state.

Since the primary oscillator is still enabled in doze mode, the device will wake up much faster and start executing instructions in the fewest amount of cycles of any low power mode. Doze mode will also have the highest current of any of the low power modes.

In order to enter doze mode, the following steps are required by the user software.

1. **Software writes to the flash control registers to put all flash pumps to sleep.** Flash pumps will not be disabled until all flash banks have been put to sleep.

> **Note:**
> Which registers need to be written depends on the specific TMS470M Series device being used. Please consult device-specific datasheet to determine the proper TMS470M Series flash module specification.

2. **Software writes to voltage regulator control register (VRCTL) to put the Voltage regulator in LPM1.** The voltage regulator will not be put into LPM1 until all the clock domains assert their clock stop acknowledge signals.

> **Note:**
> This only applies to devices with an internal voltage regulator. Consult the device specific TMS470M Series device datasheet.

3. **Software writes to the clock source disable register (CSDIS) to disable the PLL clock source.** Other sources may also be powered off, but the high-frequency oscillator must still be enabled to be considered to be in doze Mode.

> **Note:**
> The source will not actually be disabled until the clock domains that use them are disabled (see the clock Definition section of the TMS470M Series Architecture specification Section 1.5.2).

4. **Software writes to the clock domain disable register (CDDIS) to disable the GCLK (CPU clock), HCLK (system clock), VCLKP (peripheral VBUS clock), VCLK2 (peripheral VBUS clock2), VCLKA1 (asynchronous peripheral VBUS clock1), and VCLKA2 (asynchronous peripheral VBUS clock2).** All these domains must be disabled in order to be considered in doze mode. A particular domain is not actually disabled until all modules using that domain assert their clockstop acknowledge signal (see the clock Definition section of the TMS470M Series Architecture specification Section 1.5.2). The RTI1CLK and RTI2CLK domains may or may not be disabled. Doze mode is normally used in conjunction with an RTI in order to wake up the device periodically from within.

> **Note:**
> Some exceptions on which domains must be disabled in doze mode may exist, which are defined in the device-specific TMS470M Series datasheet.

5. Software “idles” the ARM core, then stops the core clock.

The device will return to normal operation after a wakeup interrupt from doze mode from one of the sources mentioned in Section 1.5.4. Which clock source is used for which clock domain after wakeup is user programmable. Normally it would make sense to use the high-frequency oscillator for the GCLK, HCLK, and VCLK domains on wakeup to minimize the time to start executing instructions after wakeup from doze mode.

If the PLL is used after returning to normal operation, a longer delay may occur since the PLL needs to re-lock. The PLL can be re-locked after waking up from doze mode in parallel with performing other operations. Once the PLL locks, the software can select the PLL as the source for the clock domains.

If an interrupt is required, it can be programmed in the M3VIM for doze mode.

2.2.3 Inactive Clock Modes

An *inactive clock* mode is defined as any low power mode in which all clock sources are off. Wakeup will be slower in these modes. Inactive clock modes require an external wakeup source or power-on reset.

Inactive clock modes can wake up on the following sources:

- External interrupt sources such as GIO, CAN, and SCI or LIN wakeup interrupt sources. The pins which can be used for wakeup are defined in the device-specific TMS470M Series device datasheet.
- Power-on reset

2.2.3.1 Sleep Mode

In sleep mode all circuits are still powered, but all clock sources are disabled as shown below:

- The PLL is disabled.
- The OSC is disabled.
- The LPO clock is disabled.
- The low-frequency clock (32 KHz oscillator or other low-frequency source) is disabled.
- The RTI clock is disabled.
- The voltage regulator (VREG), if present, must be put in LPM1 state.

Since all clocks are disabled in sleep mode the current consumption of the device is limited to normal leakage currents. Since there are no internal clocks, an external event is required to wake up from sleep mode.

Sleep mode supports the following wakeup (return to normal operation) methods:

- Any asynchronous wakeup interrupt
  - An external GIO Interrupt.
  - CAN message
  - SCI or LIN message
- Power-on or system reset

To enter sleep mode, the same steps as required for snooze mode are required by the user software, except that all clock sources must be disabled in Step 4 and all clock domains must be disabled in Step 5. If any clock is still running the device is not considered to be in sleep mode.

1. **Software writes to the flash control registers to put all flash banks to sleep.** Flash banks will not be disabled until all accesses to the flash have stopped.

   Note:
   Which registers need to be written depends on the specific TMS470M Series device being used. Please consult the device-specific datasheet to determine the proper TMS470M Series flash module specification.

2. **Software writes to the flash control registers to put all flash pumps to sleep.** Flash pumps will not be disabled until all flash banks have been put to sleep.

   Note:
   Which registers need to be written depends on the specific TMS470M Series device being used. Please consult the device-specific datasheet to determine the proper TMS470M Series flash module specification.

3. **Software writes to voltage regulator control register (VRCTL) to put the Voltage regulator in LPM1.** The voltage regulator will not be put into LPM1 until all the clock domains assert their clock stop acknowledge signals.

   Note:
   This only applies to devices with an internal voltage regulator. Consult the device specific TMS470M Series device datasheet.

4. Software writes to the clock source disable register (CSDIS) to disable all clock sources.

> **Note:**
> Note: The source will not actually be disabled until the clock domains that use them are disabled.

5. **Software writes to the clock domain disable register (CDDIS) to disable all clock domains.** All these domains must be disabled in order to be considered in sleep mode. A particular domain is not actually disabled until all modules using that domain assert their clockstop acknowledge signal.

> **Note:**
> Note: Some exceptions on which domains must be disabled in sleep mode may exist, which are defined in the device-specific TMS470M Series device datasheet.

Software executes an ARM instruction with an IDLE cycle, which triggers the CPU clock to stop. The device will return to normal operation after any asynchronous wakeup interrupt. Which clock source is used for which clock domain after wakeup is user programmable (see the device-specific TMS470M Series device datasheet). Normally, it would make sense to use the high-frequency oscillator (either the crystal oscillator or high-frequency LPO) for the GCLK, HCLK, and VCLK domains on wakeup to minimize the time to start executing instructions after wakeup from sleep mode. The selected oscillator will have to start up before the clocks will be released internally.

If the PLL is used after returning to normal operation, a longer delay may occur since the PLL needs to re-lock. The PLL can be re-locked after waking up from sleep mode in parallel with performing other operations. Once the PLL locks, the software can select the PLL as the source for the clock domains.

If an interrupt is required, it can be programmed in the M3VIM for sleep mode.

2.2.4 ARM CPU Idle Mechanisms

The mechanism used to “idle” the CPU varies per ARM core. Later ARM cores have internal power management logic which should be utilized for best results.

2.2.4.1 ARM Cortex M3

The ARM Cortex M3 core has internal power management logic. The M3 has two dedicated instructions which may be used to enter low power modes: Wait for Interrupt (WFI) and Wait for Event (WFE). The TMS470M Series platform Cortex M3 based devices support only WFI.

When a WFI instruction is executed, the core will flush its pipeline, flush its write buffer, and complete all pending bus transactions. At this time, the core will indicate to the system that it may stop the core HCLK by signalling with the SLEEPING or SLEEPDEEP signals, depending on M3 NVIC register configuration. The system will perform a logical “OR” on these signals and utilize them as a clockstop acknowledge for purposes of externally stopping core clock generation.

On receipt of a wakeup request, the GCM will start the HCLK and FCLK to the M3 core. Code execution will not restart until the wakeup interrupt is received by the core. Code execution will restart in the appropriate interrupt handler.

> **Note:**
> A Cortex M3 CPU may only be awakened by an interrupt or CPU reset. This has implications on the system level use of level sensitive external wakeup signals.

Interconnect

This chapter describes the system interconnect of the Texas Instruments TMS470M Series of microcontrollers. The system interconnect is used to connect the CPU to Program memory, RAM, etc.

Topic Page

3.1 Interconnect Overview. 50
3.2 Switched Central Resource (SCR) or Bus Matrix Module (BMM) 50
3.3 Arbiter 50
3.4 Peripheral Central Resource (PCR) 51

3.1 Interconnect Overview

The interconnect system comprises the logical datapath which connects bus masters to bus slaves. In the TMS470M Series of microcontrollers there are two general types of interconnect: system interconnect and peripheral interconnect. System interconnect is used to connect the bus masters (CPU, etc.) to SRAM, non-volatile memory, and other level 2 bus slaves. Peripheral interconnect is used to connect the system interconnect to the peripherals.

3.2 Switched Central Resource (SCR) or Bus Matrix Module (BMM)

The SCR/BMM provides connectivity between different bus slave modules to different bus master modules. Accesses from different master modules are executed in parallel if no resource conflict occurs or if the master modules are kept in series through arbitration.

The basic implementation of the SCR is the multi-layer bus matrix. The TMS470M Series SCR consists of one decoder per master and one arbiter per slave. The decoder decodes accesses from different masters and the arbiter arbitrates between accesses to the same slave by different masters.

The bus matrix provides high bus bandwidth to each slave by providing a dedicated bus to each slave. The functions of the SCR are listed below.

- Allows parallel execution of ARM buses when different slaves are accessed
- Decodes accesses from ARM master ports
- Decodes accesses to the following slave frames:
  - 128-Mbyte, two RAM frame
  - 16-Mbyte peripheral frame
  - 128-Mbyte reserved two expansion RAM frame
- Generates appropriate slave response for accesses to reserved memory map regions
- Arbitrates master accesses to slaves, currently including RAM0, RAM1, Peripheral Bridge, CRC
- Supports round robin and fixed priority schemes

3.3 Arbiter

3.3.1 Arbitration Scheme

There are two programmable arbitration schemes.

- Fix priority. The order of the fixed priority is:
  1. CPU instruction bus
  2. CPU data read,
  3. CPU data write
- Round robin. The most recent master becomes the lowest priority.

The programming of the priority is done by the SCR/BMM control register 2 (BMMCR2) in the system module.

3.4 Peripheral Central Resource (PCR)

The PCR provides decoding and multiplexing between the peripheral bridge and the peripherals and memories.

The peripheral bridge is connected to the PCR using the VBUSP protocol.

The VBUSP protocol is used to communicate with the peripherals (PS0-PS31) and the peripheral memory frame (PCS0 to PCS63). The PCR also incorporate a clock management interface, for these non-system peripherals/peripheral memories. Depending on certain conditions, discussed further in section 3.4.1–section 3.4.3, PCR can turn on or power down the clocks to these peripherals individually.

Each peripheral select frame is divided into 4 quadrants that occupy 256 bytes each. The PCR has the following features:

- Allows up to 128 peripherals to be connected (4 quadrants x 32 peripheral frames).
- Allows up to 64 peripheral memories of sizes from 1K byte to 128K bytes

It is possible to restrict writes to privilege mode only to each one of the non-system peripherals individually. This protection is done by configuring the PCR_PMPROT (for memories) or the PCR_PPROT (for peripherals) registers in PCR. The protection granularity is done at each peripheral memory chip select boundary and at each peripheral quadrant boundary.

Refer to Table 1-6 for limitations and error responses of peripheral selects and peripheral memory chip selects.

Note:
Writes are allowed in debug mode, i.e., when SUSPEND = 1, even if writes are restricted to privilege mode only.

If a selected slave does not respond with the correct VBUSP ready within 64 VBUSP clocks, PCR terminates the request by de-asserting the peripheral select to the slave and asserts the timeout error back to the master (unless in debug mode), along with the appropriate READY.

3.4.1 PCR Operation

When the PCR receives a bus request from the bridge, the vbusp_address bus is decoded and the request is sent to the corresponding slave.

If an illegal address (non implemented address) is decoded within the PCR, it will be treated as a request to an external slave.If the decoding is successful, and the slave classifies it as an illegal address/ access, it communicates it to the PCR.

In case of a timeout error from the slave, the vbusp_toerror will be muxed the same way as vbusp_aerror.

If PCR asserts a request to a slave and the slave does not respond with the appropriate ready within 64 VBUSP clocks (VCLK_sys), then the PCR de-asserts the request to the slave and asserts a timeout error to the master, along with the appropriate ready signal.

In debug mode, if the 64 clock timeout occurs, PCR doesn't assert the timeout error. However, it does terminate the request and asserts the appropriate ready signal back to the master.

PCR implements two registers to protect the non-system peripherals:

- PCR_PMPROT (for PCS region)
- PCR_PPROT (for PS region).

One bit is implemented per slave. These registers are accessible as set-clear registers. Please refer to section 4.2 for a description of the PCR registers.

PCR detects privilege violation in VBUSP accesses, according to the PCR_PMPROT register or the PCR_PPROT register. If a write to a privileged location is done in user mode, but the corresponding bit in the protection registers is 1, then the PCR will return an address error to the bridge. If an illegal access is detected, the request is not propagated to the peripheral.

In debug mode, the PCR does not check for the privilege of VBUSP accesses.

3.4.2 Clock Management Interface

The PCR module drives the clock to the peripheral clock domain (VCLK_Periph). The sources for this clock are defined in the GHVSRC register (see section 4.2.18) and can be disabled via bit 2 in the CDDISxx register . The clocks to each peripherals can be turned on/off independently via the PCSPWRDWNxxx and the PSPWRDOWNxxx registers.

The clock and the three other control signals constitute the clock management interface that PCR has with a non-system peripheral.

When all of the conditions described below are satisfied, PCR initiates the procedures to turn off clocks to a peripheral:

- Reset is inactive (high).
- No VBUSP request is pending or ongoing to that peripheral.
- The local power-down bit for that peripheral is 1, or VCLK_Periph disable bit in the clock domain disable register is set.
- The Test signal is inactive (low).
- The Wake-up is not asserted

The PCR can turn off clocks to any peripheral independent of the clocks to any other peripheral.

Once all the conditions required to disable the clocks to a peripheral have been met, the following occurs:

1. The PCR asserts the appropriate Clock Stop Request signal, on the rising edge of VCLK_Periph.
2. Once the slave asserts a Clock Stop Acknowledgment signal on the rising edge of VCLK_Periph, the clocks to the specific slave being powered down are turned off, beginning from their next respective low phases.

After the clocks to a peripheral have been disabled, they can be restarted when the slave asserts its SLAVE_WAKEUP. Alternately, when one of the above-mentioned conditions becomes untrue (not satisfied), the clock is restarted.

Sometimes, it may occur that before the slave has responded with an acknowledge to a clock stop request from the PCR, one or more of the conditions listed above may not be satisfied; e.g., a VBUSP access may occur to the particular peripheral, or the power-down bit may become 0, or maybe a system reset signal arrives. When this happens, the PCR drops the clock stop request and the clocks continue running.

This description applies for each of the non-system peripherals. This also applies, with a few exceptions described below, to the external slave, if one is defined.

If an external slave is defined, because no power-down bit is associated with that peripheral, the external peripheral clock will be disabled only if the VBUS_sys clock is disabled. Since the clock used for the external slave is VBUSP_sys, the same clock clocks the bus. This occurs because this external clock is used to drive another PCR, which is by definition a system module.

Unlike other peripherals, only one VBUSP clock is driven out for the external slave. There is no nVBUSP (inverted VBUSP) clock for the external peripheral. It is assumed that the external nVBUSP clock will be derived off-chip from the VBUSP clock that the PCR drives out.

If an external slave is not defined, the external clock is permanently low. The associated clock stop request is always 0.

Not every peripheral select (PS) or peripheral memory select (PCS) has its own clock. Some peripherals share buses with associated memories (e.g., MIBSPI and MIBSPI_RAM). In these cases, similar to the case of the VBUSP-related signals, a peripheral has only one clock management interface with the PCR.

3.4.3 Low Power Mode Interface

When the VBUSP_periph disable bit is set, the PCR asserts a clock stop request to all the slaves. After all the slaves have acknowledged the request and their clocks have been disabled, the PCR drives the power-down acknowledge active back to the GCM, indicating that the clocks to all the peripherals (VBUSP_periph) have been disabled.

If any one of the slaves asserts its Slave Wake-up signal, the PCR Wake-up signal is asserted back to the GCM, asynchronously. This assertion won't cause any glitches because the clocks are low when stopped. When this assertion occurs, all the clocks in the system are re-enabled.

The power-down acknowledge signal and the PCR Wake-up signal are made inactive, when the VBUSP_periph clock domain is enabled.

When PCR has asserted the power-down acknowledge signal, if any peripheral needs to be awakened because of either a system reset or a VBUSP access, the PCR Wake-up signal is asserted to the GCM to re-enable the clock domain. All clocks are stopped in their low phases, in a glitch-free manner.

Control Registers

This chapter describes the control registers of the Texas Instruments TMS470M Series system architecture.

Topic **Page**

4.1 System Peripherals Memory Mapping 56
4.2 System Control Registers (SYS) 58
4.3 Peripheral Central Resource (PCR) Control Registers 140
4.4 System Control Registers Secondary Frame(SYS2) 174

4.1 System Peripherals Memory Mapping

The SYSTEM peripheral frame regroups all the system-sensitive registers and system memory frames.

Note:
For device specific implementation details, reference the device-specific datasheet .

Table 4-1. SYSTEM Peripheral Frames

Frame Name	Quadrant	Peripheral Name	Start Address	End Address	Size in Bytes	Data Access
PPCS0	N/A	Reserved	0xFFF8 0000	0xFFF8 0FFF	4 K	8/16/32
PPCS1	N/A	Reserved	0xFFF8 1000	0xFFFF 1FFF	4 K	8/16/32
PPCS2	N/A	VIM RAM	0xFFF8 2000	0xFFF8 2FFF	4 K	8/16/32
PPCS3	N/A	Reserved	0xFFF8 3000	0xFFF8 3FFF	4 K	8/16/32
PPCS4	N/A	Reserved	0xFFF8 4000	0xFFF8 4FFF	4 K	8/16/32
PPCS5	N/A	Reserved	0xFFF8 5000	0xFFF8 5FFF	4 K	8/16/32
PPCS6	N/A	Reserved	0xFFF8 6000	0xFFF8 6FFF	4 K	8/16/32
PPCS7	N/A	Flash wrapper 1	0xFFF8 7000	0xFFF8 7FFF	4 K	8/16/32
PPCS8	N/A	Flash Wrapper 2 (opt.)	0xFFF8 8000	0xFFF8 8FFF	4 K	8/16/32
PPCS9	N/A	ICEcrusher7E (opt.)	0xFFF8 9000	0xFFF8 9FFF	4 K	8/16/32
PPCS10	N/A	Reserved	0xFFF8 A000	0xFFF8 AFFF	4 K	8/16/32
PPCS11	N/A	Reserved	0xFFF8 B000	0xFFF8 BFFF	4 K	8/16/32
PPCS12	N/A	Reserved	0xFFF8 C000	0xFFF8 CFFF	4 K	8/16/32
PPCS13	N/A	Reserved	0xFFF8 D000	0xFFF8 DFFF	4 K	8/16/32
PPCS14	N/A	Reserved	0xFFF8 E000	0xFFF8 EFFF	4 K	8/16/32
PPCS15	N/A	Reserved	0xFFF8 F000	0xFFF8 FFFF	4 K	8/16/32
	N/A	Reserved	0xFFF9 0000	0xFFFE FFFF		
PPCS16	N/A	Reserved	0xFFFF 0000	0xFFFF 7FFF	32 K	8/16/32
	N/A	Reserved	0xFFFF 8000	0xFFFF DFFF		
PPS0	0	PCR Registers	0xFFFF E000	0xFFFF E0FF	256	8/16/32
	1	System Frame 2 Regs	0xFFFF E100	0xFFFF E1FF	256	8/16/32
	2	RAM ECC Regs 3	0xFFFF E200	0xFFFF E2FF	256	8/16/32
	3	RAM ECC Regs 4	0xFFFF E300	0xFFFF E3FF	256	8/16/32
PPS1	0	Memory BISTcontroller 1 (device-specific)	0xFFFF E400	0xFFFF E4FF	256	8/16/32
	1	Memory BIST controller 2 (device-specific)	0xFFFF E500	0xFFFF E5FF	256	8/16/32
	2	Logic BIST controller 1 (device-specific)	0xFFFF E600	0xFFFF E6FF	256	8/16/32
	3	Logic BIST controller 2 (device-specific)	0xFFFF E700	0xFFFF E7FF	256	8/16/32
PPS2	0	EMIF 1 Registers (opt.)	0xFFFF E800	0xFFFF E8FF	256	8/16/32
	1	EMIF 2 Registers (opt.)	0xFFFF E900	0xFFFF E9FF	256	8/16/32
	2	Reserved	0xFFFF EA00	0xFFFF EAFF	256	8/16/32
	3	Reserved	0xFFFF EB00	0xFFFF EBFF	256	8/16/32

Table 4-1. SYSTEM Peripheral Frames (Continued)

Frame Name	Quadrant	Peripheral Name	Start Address	End Address	Size in Bytes	Data Access
PPS3	0	Reserved	0xFFFF EC00	0xFFFF ECFF	256	8/16/32
	1	Wakeup Registers	0xFFFF ED00	0xFFFF EDFF	256	8/16/32
	2	RTI2 Registers (opt.)	0xFFFF EE00	0xFFFF EEFF	256	8/16/32
	3	Pin Mux control (opt.)	0xFFFF EF00	0xFFFF EFFF	256	8/16/32
PPS4	0	Reserved	0xFFFF F000	0xFFFF F0FF	256	8/16/32
	1		0xFFFF F100	0xFFFF F1FF	256	8/16/32
	2		0xFFFF F200	0xFFFF F2FF	256	8/16/32
	3		0xFFFF F300	0xFFFF F3FF	256	8/16/32
PPS5	0	Reserved	0xFFFF F400	0xFFFF F4FF	256	8/16/32
	1	ESM Registers	0xFFFF F500	0xFFFF F5FF	256	8/16/32
	2	CCM7 Register (opt.)	0xFFFF F600	0xFFFF F6FF	256	8/16/32
	3	DMM Registers (opt.)	0xFFFF F700	0xFFFF F7FF	256	8/16/32
PPS6	0	RAM ECC Regs 2 (opt.)	0xFFFF F800	0xFFFF F8FF	256	8/16/32
	1	RAM ECC Regs 1 (opt.)	0xFFFF F900	0xFFFF F9FF	256	8/16/32
	2	Reserved	0xFFFF FA00	0xFFFF FAFF	256	8/16/32
	3	Reserved	0xFFFF FB00	0xFFFF FBFF	256	8/16/32
PPS7	0	RTI	0xFFFF FC00	0xFFFF FCFF	256	8/16/32
	1	VIM Parity Register (opt.)	0xFFFF FD00	0xFFFF FDFF	256	8/16/32
	2	M3VIM Registers	0xFFFF FE00	0xFFFF FEFF	256	8/16/32
	3	System module Registers	0xFFFF FF00	0xFFFF FFFF	256	8/16/32

4.2 System Control Registers (SYS)

This section describes the SYSTEM registers. There are two frames available for registers. The start address of the first system module frame is 0xFFFF FF00 and the end address is 0xFFFF FFFF. The start address of the second system module frame is 0xFFFF E100 and the end address is 0xFFFF E1FF. The registers support 32-, 16-, and 8-bit writes. The offset is relative to the system module frame start address.

Figure 4-1 contains a summary figure of the system control registers.

Figure 4-1. Frame 1 System Control Registers Summary

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
0x00 SYSPC1 Page 66	Reserved															
	Reserved														ECP-CLK2FUN	ECP-CLK-FUN
0x04 SYSPC2 Page 67	Reserved															
	Reserved														ECP CLK2 DIR	ECP CLK DIR
0x08 SYSPC3 Page 68	Reserved															
	Reserved														ECP CLK2 DIN	ECP CLK DIN
0x0C SYSPC4 Page 69	Reserved															
	Reserved														ECP CLK2 DOUT	ECP CLK DOUT
0x10 SYSPC5 Page 71	Reserved															
	Reserved														ECP CLK2 SET	ECP CLK SET
0x14 SYSPC6 Page 72	Reserved															
	Reserved														ECP CLK2 CLR	ECP CLK CLR
0x18 SYSPC7 Page 73	Reserved															
	Reserved														ECP CLK2 ODE	ECP CLK ODE

Figure 4-1. Frame 1 System Control Registers Summary (Continued)

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
0x1C SYSPC8 Page 75	Reserved															
	Reserved														ECP CLK2 PUE	ECP CLK PUE
0x20 SYSPC9 Page 76	Reserved															
	Reserved														ECP CLK2 PS	ECP CLK PS
0x24 SSWPLL1 Page 77	Reserved															
	MOD_PH_CAP_INDEX(7:0)								Reserved	COUNTER_READ_READY	COUNTER_RESET	COUNTER_EN	TAP_COUNTER_DIS(2–1)			EXT_COUNTER_EN
0x28 SSWPLL2 Page 79	SSW_CAPTURE_COUNT(31–16)															
	SSW_CAPTURE_COUNT(16–0)															
0x2C SSWPLL3 Page 80	SSW_CLKOUT_COUNT(31–16)															
	SSW_CLKOUT_COUNT(16–0)															
0x30 CSDIS Page 81	Reserved															
	Reserved								CLKSR7OFF	CLKSR6OFF	CLKSR5OFF	CLKSR4OFF	CLKSR3OFF	CLKSR2OFF	CLKSR1OFF	CLKSR0OFF
0x34 CSDISSET Page 82	Reserved															
	Reserved								CLKSR7OFFSET	CLKSR6OFFSET	CLKSR5OFFSET	CLKSR4OFFSET	CLKSR3OFFSET	CLKSR2OFFSET	CLKSR1OFFSET	CLKSR0OFFSET
0x38 CSDISCLR Page 83	Reserved															
	Reserved								CLKSR7OFFCLR	CLKSR6OFFCLR	CLKSR5OFFCLR	CLKSR4OFFCLR	CLKSR3OFFCLR	CLKSR2OFFCLR	CLKSR1OFFCLR	CLKSR0OFFCLR

Figure 4-1. Frame 1 System Control Registers Summary (Continued)

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
0x3C CDDIS Page 84	Reserved															
	Reserved								RTI CLK2 OFF	RTI CLK1 OFF	VCLK A 2 OFF	VCLK A OFF	VCLK 2 OFF	VCLK P OFF	HCLK OFF	GCLK OFF
0x40 CSDISSET Page 85	Reserved															
	Reserved								RTI CLK2 OFF SET	RT CLK1 OFF SET	VCLK A2 OFF SET	VCLK AOFF SET	VCLK 2OFF SET	VCLK P OFF SET	HCLK OFF SET	GCLK OFF SET
0x44 CSDISCLR Page 87	Reserved															
	Reserved								RTI CLK2 OFF CLR	RTI CLK1 OFF CLR	VCLK A2 OFF CLR	VCLK AOFF CLR	VCLK 2OFF CLR	VCLK P OFF CLR	HCLK OFF CLR	GCLK ENA CLR
0x48 GHVSRC Page 89	Reserved								GHVWAKE(3:0)				Reserved			
	Reserved												GHVSRC(3:0)			
0x4C VCLKASRC Page 91	Reserved															
	Reserved								VCLKA2S(3:0)				Reserved			
													VCLKA1S(3:0)			
0x50 RCLKSRC Page 93	Reserved								RTI2DIV(1:0)				Reserved			
	Reserved								RTI1DIV(1:0)				Reserved			
													RTI1SRC(3:0)			
0x54 CSVSTAT Page 95	Reserved															
	Reserved								CLK SR7V	CLK SR6V	CLK SR5V	CLK SR4V	CLK SR3V	CLK SR2V	CLK SR1V	CLK SR0V
0x58 MSTGCR Page 96	Reserved															
	Reserved								ROM_DIV(1:0)				Reserved			
													MSTGENA(3:0)			

Figure 4-1. Frame 1 System Control Registers Summary (Continued)

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
0x5C MINITGCR Page 98	Reserved															
	Reserved												MINITGENA(3:0)			
0x60 MSIENA Page 99	MSIENA[31:16]															
	MSIENA[15:0]															
0x64 MSTFAIL Page 100	MSTF[31:16]															
	MSTF[15:0]															
0x68 MSINIGSTAT Page 101	Reserved															
	Reserved							MINI DONE	Reserved							MST DONE
0x6C MINISTAT Page 102	MIDONE[31:16]															
	MIDONE[15:0]															
0x70 PLLCTL1 Page 103	Device Specific															
	Device Specific															
0x74 PLLCTL2 Page 103	Device Specific															
	Device Specific															
0x78 Reserved	Reserved															
	Reserved															

Figure 4-1. Frame 1 System Control Registers Summary (Continued)

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
0x7C DIEIDL Page 104	DIEIDL(31:16)															
	DIEIDL(15:0)															
0x80 DIEIDH Page 105	DIEIDH(31:16)															
	DIEIDH(15:0)															
0x84 VRCTL Page 106	Reserved															
	Reserved										VLPMENA(3:0)				VSLEEPENA(3:0)	
0x88 LPOMONCTL Page 108	Reserved							BIAS ENA BLE	Reserved							
	Reserved				HFTRM(3:0)				Reserved				LFTRM(3:0)			
0x8C CLKTEST Page 110	Reserved					RANG EDE-TCTR L	RANG EDE-TEN-ASSE L	Reserved					CLK_TEST_EN(3:0)			
	Reserved				SEL_GIO_PIN(3:0)				Reserved				SEL_ECP_PIN(3:0)			
0x90-0x9C Reserved	Reserved															
	Reserved															
0xA0 GPREG1 Page 113	Device Specific															
	Device Specific															
0xA4 BTRMSEL Page 114	Reserved															
	Reserved															

Figure 4-1. Frame 1 System Control Registers Summary (Continued)

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
0xA8 IMPFASTS Page 115	Reserved								MASTERID							
	Reserved							EMIFA	VBUSA	Reserved						ATYPE
0xAC IMPFTADD Page 117	IMPFTADD(31:16)															
	IMPFTADD(15:0)															
0xB0 SSIR1 Page 118	Reserved															
	SSKEY1								SSDATA1							
0xB4 SSIR2 Page 119	Reserved															
	SSKEY2								SSDATA2							
0xB8 SSIR3 Page 120	Reserved															
	SSKEY3								SSDATA3							
0xBC SSIR4 Page 121	Reserved															
	SSKEY4								SSDATA4							
0xC0 RAMGCR Page 122	Reserved												RAM_DFT_EN(3:0)			
	Reserved	WST_AENA3	Reserved	WST_DENA3	Reserved	WST_AENA2	Reserved	WST_DENA2	Reserved	WST_AENA1	Reserved	WST_DENA1	Reserved	WST_AENA0	Reserved	WST_DENA0
0xC4 BMMCR1 Page 124	Reserved															
	Reserved												MEMSW			

Figure 4-1. Frame 1 System Control Registers Summary (Continued)

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
0xC8 BMMCR2 Page 125	Reserved															
	Reserved									PRTY_)HPI	PRTY_RAM3	PRTY_RAM2	PRTY_CRC	PRTY_PBRG	PRTY_RAM1	PRTY_RAM0
0xCC MMUGCR Page 127	Reserved															
	Reserved															MPM ENA
0xD0 CLKCNTL Page 128	Reserved					VCLK2R[3:0]			Reserved				VCLKR[3:0]			
	Reserved							PENA	Reserved							
0xD4 ECPCNTRL Page 129	Reserved							ECP-SSEL	ECP COS	Reserved						
	ECPDIV															
0xD8 DSPGCR Page 130	Reserved															
	Reserved															
0xDC DEVCR1 Page 131	Reserved															
	Reserved												DEVPARSEL			
0xE0 SYSECR Page 132	Reserved															
	RESET[1]	RESET[0]	Reserved													
0XE4 SYSESR Page 133	Reserved															
	PO RST	OSC RST	WD RST	Reserved							CPU RST	SW RST	EXT RST	Reserved		

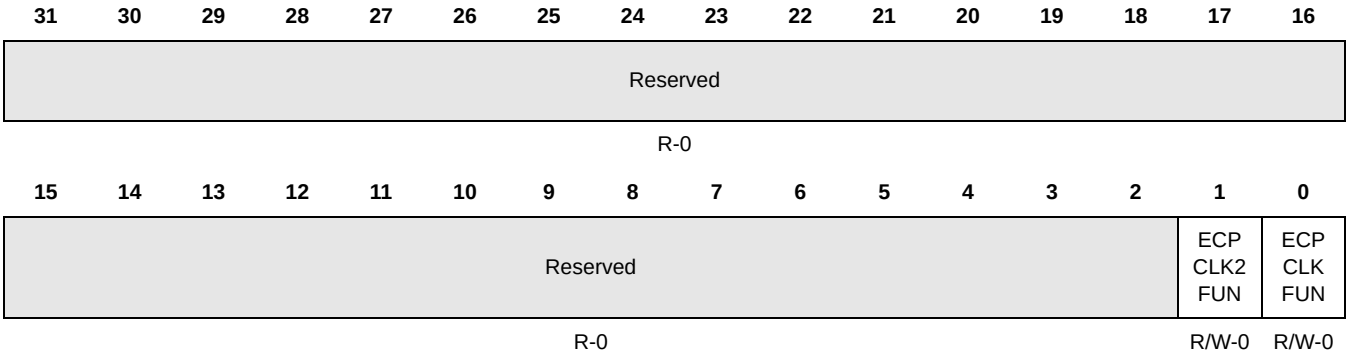
Figure 4-1. Frame 1 System Control Registers Summary (Continued)

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
0xE8 Reserved	Reserved															
	Reserved															
0xEC GLBSTAT Page 135	Reserved															
	Reserved															OSC-FAIL
0xF0 DEVID Page 136	CP15	ID														TECH
	TECH			I/O	PPAR	Program parity		RECC	Version					1	0	1
0xF4 SSIVEC Page 138	Reserved															
	SSIDATA								SSIVECT[7:0]							
0xF8 SSIF Page 139	Reserved															
	Reserved												SSI_FLAG 4	SSI_FLAG 3	SSI_FLAG 2	SSI_FLAG 1
0xFC SSIR1 Page 118	Reserved															
	SSKEY1								SSDATA1							

4.2.1 SYS Pin Control Register 1 (SYSPC1)

The SYSPC1 register, shown in Figure 4-2 and described in Table 4-2, controls the function of the ECPCLK and ECPCLK2.

Figure 4-2. SYS Pin Control Register 1 (SYSPC1) [offset = 0x00]



R = Read in all modes; W = write in all modes; n = value after reset

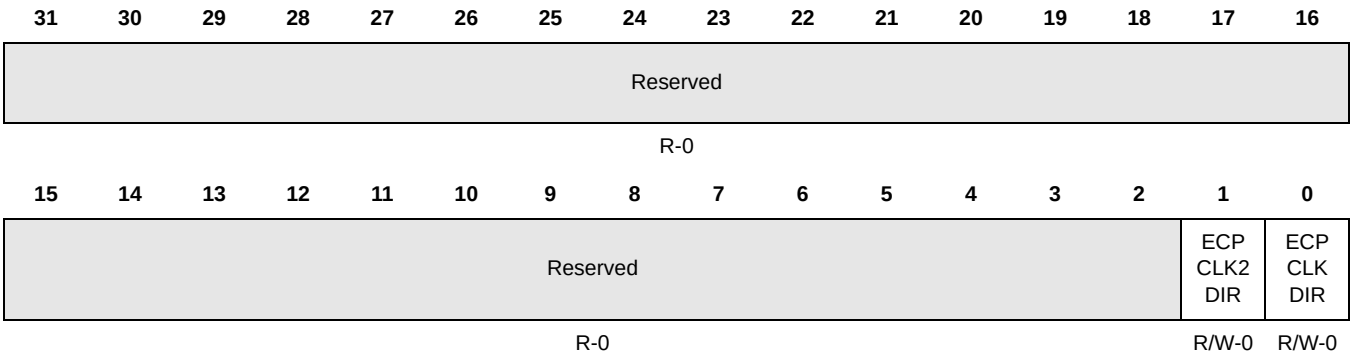
Table 4-2. SYS Pin Control Register 1 (SYSPC1) Field Descriptions

Bit	Name	Value	Description
31–2	Reserved		Reads return zero and writes have no effect.
1	ECPCLK2FUN		ECPCLK2 function. This bit changes the function of the ECPCLK2 .
		0	ECPCLK2 is in GIO mode.
		1	ECPCLK2 is in functional mode as an output.
			Note: **This bit controls ECPCLK2. ECPCLK2 is muxed with HET15. This feature/bit is only available on some configurations of TMS470M series. Please check the device specific datasheet for applicability to the specific part number being used.**
0	ECPCLKFUN		ECPCLK function. This bit changes the function of the ECPCLK.
		0	ECPCLK is in GIO mode.
		1	ECPCLK is in functional mode as an output.
			Note: **It is recommended not to disable and enable the ECPCLK in functional mode within the same ECP clock period to guarantee a proper ECP clock duty cycle.**

4.2.2 SYS Pin Control Register 2 (SYSPC2)

The SYSPC2 register, shown in Figure 4-3 and described in Table 4-3, controls the function of the ECPCLK and ECPCLK2 pin.

Figure 4-3. SYS Pin Control Register 2 (SYSPC2) [offset = 0x04]



R = Read; W = Write; -n = value after reset

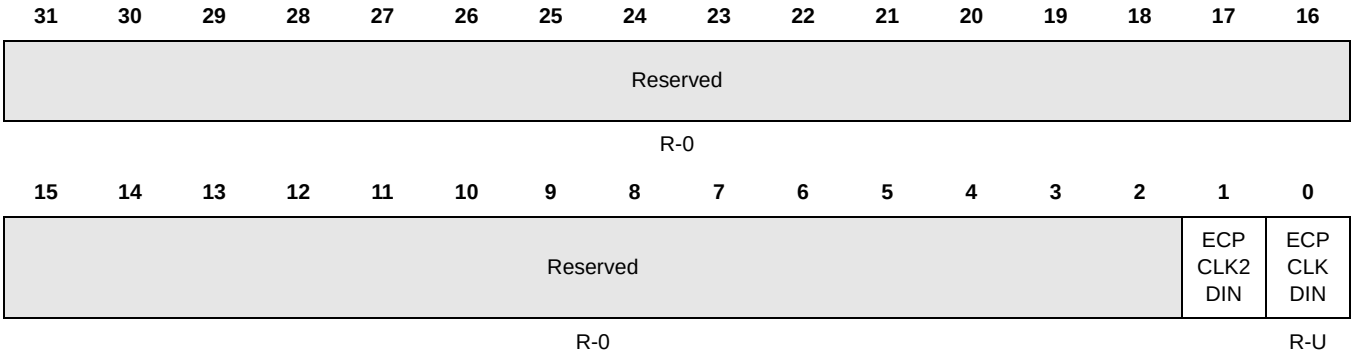
Table 4-3. SYS Pin Control Register 2 (SYSPC2) Field Descriptions

Bit	Name	Value	Description
31–2	Reserved		Reads return zero and writes have no effect.
1	ECPCLK2_DIR		ECPCLK2 data direction. This bit controls the direction of the ECPCLK2 pin when it is configured to be in GIO mode only (ECPCLK2FUN = 0).
		0	The ECPCLK2 pin is an input.
		1	The ECPCLK2 pin is an output.
			Note: **This bit controls ECPCLK2. ECPCLK2 is muxed with HET15. This feature/bit is only available on some configurations of TMS470M series. Please check the device specific datasheet for applicability to the specific part number being used.**
0	ECPCLK_DIR		ECPCLK data direction. This bit controls the direction of the ECPCLK pin when it is configured to be in GIO mode only (ECPCLKFUN = 0).
		0	The ECPCLKGIO pin is an input.
		1	The ECPCLKGIO pin is an output.

4.2.3 SYS Pin Control Register 3 (SYSPC3)

The SYSPC3 register, shown in Figure 4-4 and described in Table 4-4, controls the function of the ECPCLK and ECPCLK2 pin.

Figure 4-4. SYS Pin Control Register 3 (SYSPC3) [offset = 0x08]



R = Read only; -n = value after reset; -U = undefined

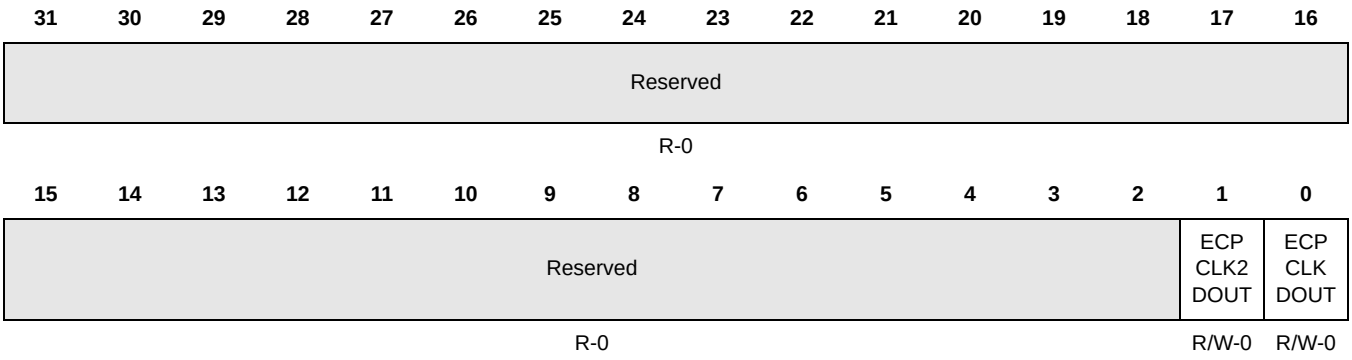
Table 4-4. SYS Pin Control Register 3 (SYSPC3) Field Descriptions

Bit	Name	Value	Description
31–2	Reserved		Reads return zero and writes have no effect.
1	ECPCLK2_DIN		ECPCLK2 data in.
		0	The ECPCLK2 pin is at logic low (0).
		1	The ECPCLK2 pin is at logic high (1).
			Note: **This bit controls ECPCLK2. ECPCLK2 is muxed with HET15. This feature/bit is only available on some configurations of TMS470M series. Please check the device specific datasheet for applicability to the specific part number being used.**
0	ECPCLK_DIN		ECPCLK data in.
		0	The ECPCLK pin is at logic low (0).
		1	The ECPCLK pin is at logic high (1).

4.2.4 SYS Pin Control Register 4 (SYSPC4)

The SYSPC2 register, shown in Figure 4-5 and described in Table 4-5, controls the function of the ECPCLK and ECPCLK2 pin output.

Figure 4-5. SYS Pin Control Register 4 (SYSPC4) [offset = 0x0C]



R = Read; W = Write; -n = value after reset

Table 4-5. SYS Pin Control Register 4 (SYSPC4) Field Descriptions

Bit	Name	Value	Description
31–2	Reserved		Reads return zero and writes have no effect.
1	ECPCLK2_DOUT		ECPCLK2 data out write. This bit is only active when ECPCLK2 is configured to be in GIO mode (ECPCLK2FUN = 0) and configured to be an output pin (ECPCLK2_DIR = 1). The value of this bit indicates the value to be output to the ECPCLK2 pin.
		0	The ECPCLK2_DOUT pin is at logic low (0). NOTE: If ECPCLK2_DIR = 1, writing a 0 to ECPCLK2_DOUT will also set ECPCLK2 to 0.
		1	The ECPCLK2_DOUT pin is at logic high (1). NOTE: If EPCLK2_DIR = 1, writing a 1 to ECPCLK2_DOUT will also set ECPCLK2 to 1.
			Note: **This bit controls ECPCLK2. ECPCLK2 is muxed with HET15. This feature/bit is only available on some configurations of TMS470M series. Please check the device specific datasheet for applicability to the specific part number being used.**
0	ECPCLK_DOUT		ECPCLK data out write. This bit is only active when ECPCLK is configured to be in GIO mode (ECPCLKFUN = 0) and configured to be an output pin (ECPCLK_DIR = 1). The value of this bit indicates the value to be output to the ECPCLK pin.
		0	The ECPCLK_DOUT pin is at logic low (0). NOTE: If ECPCLK_DIR = 1, writing a 0 to ECPCLK_DOUT will also set ECPCLK to 0.

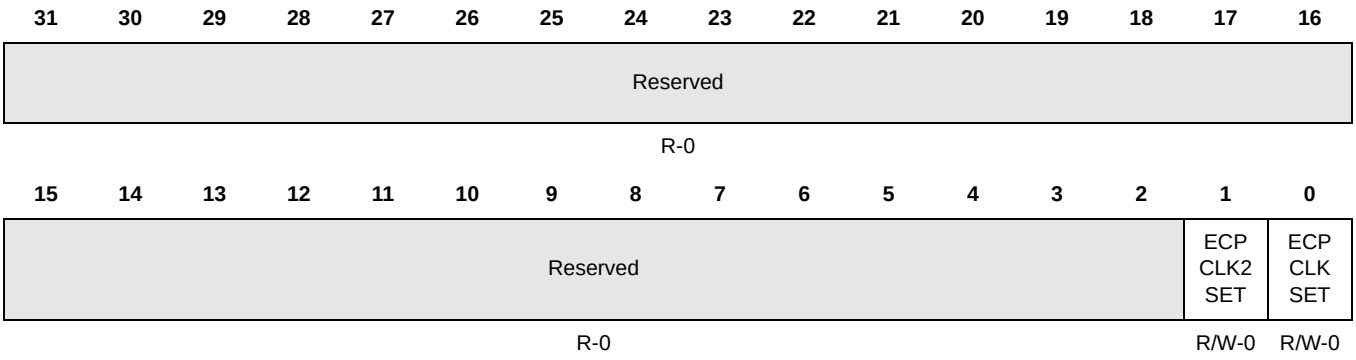
Table 4-5. SYS Pin Control Register 4 (SYSPC4) Field Descriptions (Continued)

Bit	Name	Value	Description
		1	The ECPCLK_DOUT pin is at logic high (1). NOTE: If ECPCLK_DIR = 1, writing a 1 to ECPCLK_DOUT will also set ECPCLK to 1.

4.2.5 SYS Pin Control Register 5 (SYSPC5)

The SYSPC5 register, shown in Figure 4-6 and described in Table 4-6, controls the function of the ECPCLK and ECPCLK2 pin output.

Figure 4-6. SYS Pin Control Register 5 (SYSPC5) [offset = 0x10]



R = Read; W = Write; -n = value after reset

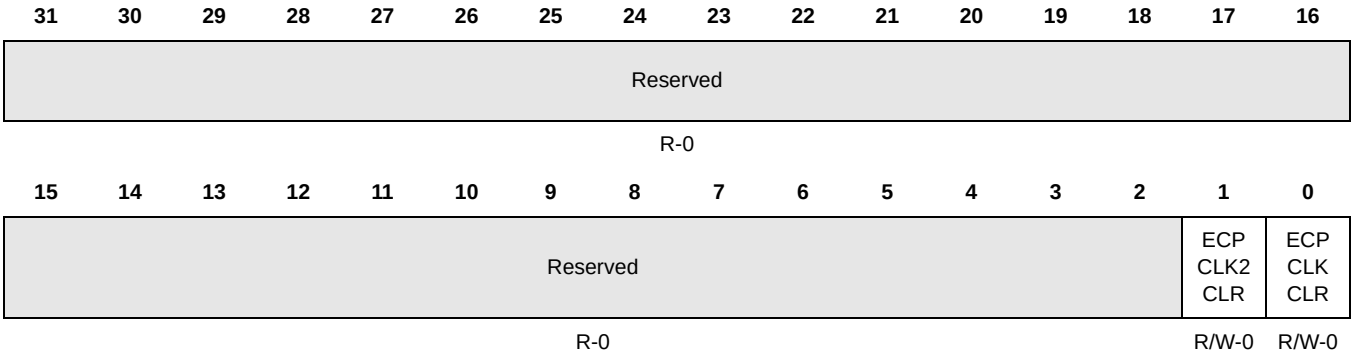
Table 4-6. SYS Pin Control Register 5 (SYSPC5) Field Descriptions

Bit	Name	Value	Description
31–2	Reserved		Reads return zero and writes have no effect.
1	ECPCLK2_SET		ECPCLK2 data out set. This bit is only active when ECPCLK2 is configured to be in GIO mode (ECPCLK2FUN = 0).
		0	Read: The ECPCLK2_DOUT pin is at logic low (0). Write: Writing a 0 has no effect.
		1	Read: The ECPCLK2_DOUT pin is at logic high (1). Write: The ECPCLK2 pin is at logic high (1).
			Note: **This bit controls ECPCLK2. ECPCLK2 is muxed with HET15. This feature/bit is only available on some configurations of TMS470M series. Please check the device specific datasheet for applicability to the specific part number being used.**
0	ECPCLK_SET		ECPCLK data out set. This bit is only active when ECPCLK is configured to be in GIO mode (ECPCLKFUN = 0).
		0	Read: The ECPCLK_DOUT pin is at logic low (0). Write: Writing a 0 has no effect.
		1	Read: The ECPCLK_DOUT pin is at logic high (1). Write: The ECPCLK pin is at logic high (1).

4.2.6 SYS Pin Control Register 6 (SYSPC6)

The SYSPC6 register, shown in Figure 4-7 and described in Table 4-7, controls the function of the ECPCLK and ECPCLK2 pin output.

Figure 4-7. SYS Pin Control Register 6 (SYSPC6) [offset = 0x14]



R = Read; W = Write; -n = value after reset

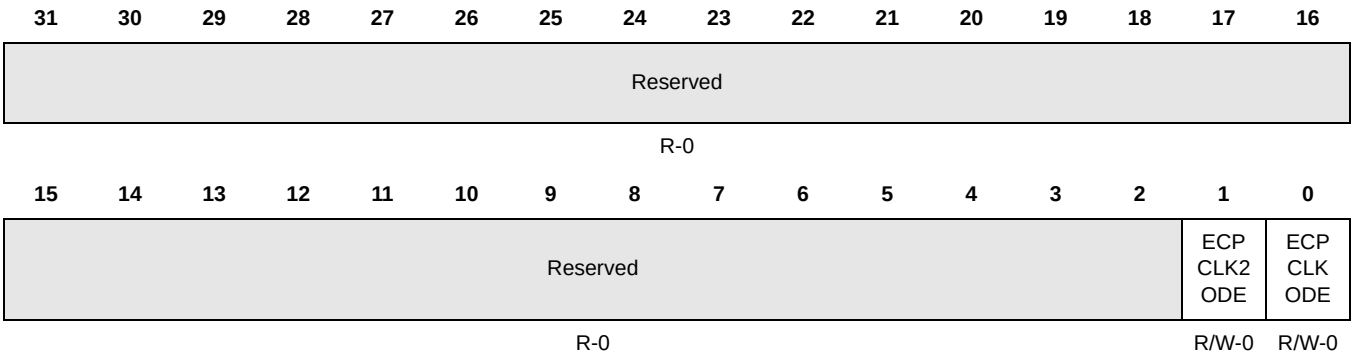
Table 4-7. SYS Pin Control Register 6 (SYSPC6) Field Descriptions

Bit	Name	Value	Description
31–2	Reserved		Reads return zero and writes have no effect.
1	ECPCLK2_SET		ECPCLK2 data out clear. This bit is only active when ECPCLK2 is configured to be in GIO mode (ECPCLK2FUN = 0).
		0	Read: The ECPCLK2_DOUT pin is at logic low (0). Write: The ECPCLK2 pin value is unchanged.
		1	Read: The ECPCLK2_DOUT pin is at logic high (1). Write: The ECPCLK2 pin is cleared to logic low (0).
			Note: **This bit controls ECPCLK2. ECPCLK2 is muxed with HET15. This feature/bit is only available on some configurations of TMS470M series. Please check the device specific datasheet for applicability to the specific part number being used.**
0	ECPCLK_SET		ECPCLK data out clear. This bit is only active when ECPCLK is configured to be in GIO mode (ECPCLKFUN = 0).
		0	Read: The ECPCLK_DOUT pin is at logic low (0). Write: The ECPCLK pin value is unchanged.
		1	Read: The ECPCLK_DOUT pin is at logic high (1). Write: The ECPCLK pin is cleared to logic low (0).

4.2.7 SYS Pin Control Register 7 (SYSPC7)

The SYSPC7 register, shown in Figure 4-8 and described in Table 4-8, controls the function of the ECPCLK and ECPCLK2 pin open drain mode.

Figure 4-8. SYS Pin Control Register 7 (SYSPC7) [offset = 0x18]



R = Read; W = Write; -n = value after reset

Table 4-8. SYS Pin Control Register 7 (SYSPC7) Field Descriptions

Bit	Name	Value	Description
31–2	Reserved		Reads return zero and writes have no effect.
1	ECPCLK2_ODE		ECPCLK2 open drain enable. This bit is only active when ECPCLK2 is configured to be in GIO mode (ECPCLK2FUN = 0).
		0	The ECPCLK2 pin is configured in push/pull mode.
		1	The ECPCLK2 pin is configured in open drain mode.
			Note: When a pin is configured as an output in GIO mode, the output buffer is tri-stated when the ECPCLK2_DOUT register is set to 1. If the ECPCLK2_DOUT register is set to 0, the output buffer drives a value of 0. If the pin direction is set as an input, the output buffer is tri-stated no matter what the value of the ECPCLK2_DOUT register is. In push/pull mode, when a pin is configured as an output in GIO mode, the pin follows whatever value is set in the DOUT register.
			Note: **This bit controls ECPCLK2. ECPCLK2 is muxed with HET15. This feature/bit is only available on some configurations of TMS470M series. Please check the device specific datasheet for applicability to the specific part number being used.**
0	ECPCLK_ODE		ECPCLK open drain enable. This bit is only active when ECPCLK is configured to be in GIO mode (ECPCLKFUN = 0).
		0	The ECPCLK pin is configured in push/pull mode.

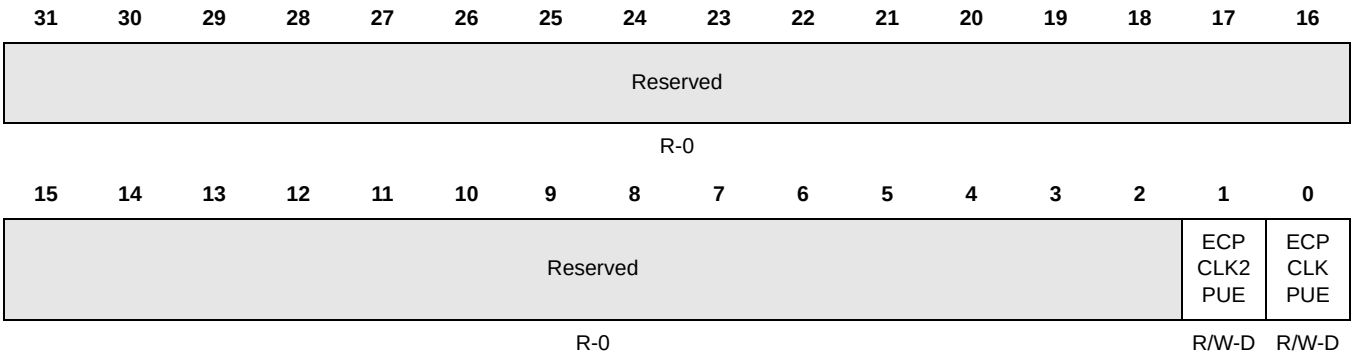
Table 4-8. SYS Pin Control Register 7 (SYSPC7) Field Descriptions (Continued)

Bit	Name	Value	Description
		1	The ECPCLK pin is configured in open drain mode. Note: When a pin is configured as an output in GIO mode, the output buffer is tri-stated when the ECPCLK_DOUT register is set to 1. If the ECPCLK_DOUT register is set to 0, the output buffer drives a value of 0. If the pin direction is set as an input, the output buffer is tri-stated no matter what the value of the ECPCLK_DOUT register is. In push/pull mode, when a pin is configured as an output in GIO mode, the pin follows whatever value is set in the DOUT register.

4.2.8 SYS Pin Control Register 8 (SYSPC8)

The SYSPC8 register, shown in Figure 4-9 and described in Table 4-9, controls the function of the ECPCLK and ECPCLK2 pin pull enable.

Figure 4-9. SYS Pin Control Register 8 (SYSPC8) [offset = 0x1C]



R = Read; W = Write; -n = value after reset; D = Device Specific

Table 4-9. SYS Pin Control Register 8 (SYSPC8) Field Descriptions

Bit	Name	Value	Description
31–2	Reserved		Reads return zero and writes have no effect.
1	ECPCLK2_PUE		ECPCLK2 pull up enable. This bit is only active when ECPCLK2 is configured to be an input.
		0	ECPCLK2 pull enable is active.
		1	ECPCLK2 pull enable is inactive.
			Note: **This bit controls ECPCLK2. ECPCLK2 is muxed with HET15. This feature/bit is only available on some configurations of TMS470M series. Please check the device specific datasheet for applicability to the specific part number being used.**
0	ECPCLK_PUE		ECPCLK pull up enable. This bit is only active when ECPCLK is configured to be an input.
		0	ECPCLK pull enable is active.
		1	ECPCLK pull enable is inactive.

4.2.9 SYS Pin Control Register 9 (SYSPC9)

The SYSPC9 register, shown in Figure 4-10 and described in Table 4-10, controls the function of the ECPCLK and ECPCLK2 pin pull up/pull down.

Figure 4-10. SYS Pin Control Register 9 (SYSPC9) [offset = 0x20]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														ECPCLK2PS	ECPCLKPS
R-0														R/W-0	R/W-0

R = Read; W = Write; -n = value after reset

Table 4-10. SYS Pin Control Register 9 (SYSPC9) Field Descriptions

Bit	Name	Value	Description
31–2	Reserved		Reads return zero and writes have no effect.
1	ECPCLK2_PS		ECPCLK2 pull up/pull down select. This bit is only active when ECPCLK2 is configured to be an input.
		0	ECPCLK2 pull down is selected, when pull up/pull down logic is enabled (ECPCLK2_PUE = 0).
		1	ECPCLK2 pull up is selected, when pull up/pull down logic is enabled (ECPCLK2_PUE = 0).
			Note: **This bit controls ECPCLK2. ECPCLK2 is muxed with HET15. This feature/bit is only available on some configurations of TMS470M series. Please check the device specific datasheet for applicability to the specific part number being used.**
0	ECPCLK_PS		ECPCLK pull up/pull down select. This bit is only active when ECPCLK is configured to be an input.
		0	ECPCLK pull down is selected, when pull up/pull down logic is enabled (ECPCLK_PUE = 0).
		1	ECPCLK pull up is selected, when pull up/pull down logic is enabled (ECPCLK_PUE = 0).

4.2.10 SSW PLL BIST Control Register 1 (SSWPLL1)

The SSWPLL1 register is shown in Figure 4-11 and described in Table 4-11. This register is for TI internal use only.

Figure 4-11. SSW PLL BIST Control Register 1 (SSWPLL1) [offset = 0x24]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MOD_PH_CAP_INDEX(7:0)								Reserved	COUNTER_READ_READY	COUNTER_RESET	COUNTER_EN	TAP_COUNTER_DIS			EXT_COUNTER_EN
R/W-0									R/W-0	R/W-1	R/W-0	R/W-0			R/W-0

R = Read; W = Write; -n = value after reset

Table 4-11. SSW PLL BIST Control Register 1 (SSWPLL1)) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Reads return zero and writes have no effect.
15–8	MOD_PH_CAP_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.
7	Reserved		Reads return zero and writes have no effect.
6	COUNTER_READ_READY		Counter read ready. If EXT_COUNTER_EN is 1 the COUNTER_READ_READY signal becomes 1 whenever COUNTER_EN bit is 0.
		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.
		0	Depending on COUNTER_EN bit, the counter value will increment or maintain its value.
		1	If the EXT_COUNTER_EN bit is 0,then counters 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.
		0	The counters are disabled (stop counting) if EXT_COUNTER_EN is 1.
		1	The counters are enabled if EXT_COUNTER_EN is 1. If EXT_COUNTER_EN bit is 1 then counters will reset on the rising edge of this bit.

Table 4-11. SSW PLL BIST Control Register 1 (SSWPLL1)) Field Descriptions (Continued)

Bit	Name	Value	Description
3-1	TAP_COUNTER_DIS (3–0)		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.
		000	Bit 16 of CLKOUT counter is selected.
		001	Bit 18 of CLKOUT counter is selected.
		010	Bit 20 of CLKOUT counter is selected.
		011	Bit 22 of CLKOUT counter is selected.
		100	Bit 24 of CLKOUT counter is selected.
		101	Bit 26 of CLKOUT counter is selected.
		110	Bit 28 of CLKOUT counter is selected.
		111	Bit 30 of CLKOUT counter is selected.
0	EXT_COUNTER_EN		
		0	Capture counter counts at every rising edge of PLLCLK.
		1	Capture counter counts at every rising edge of oscillator input.

4.2.11 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. The SSWPLL2 register is shown in Figure 4-12 and described in Table 4-12. This register is for TI internal use only.

Figure 4-12. SSW PLL BIST Control Register 2 (SSWPLL2) [offset = 0x28]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SSW_CAPTURE_COUNT															
R-0															

R = Read; W = Write; -n = value after reset

Table 4-12. SSW PLL BIST Control Register 2 (SSWPLL2) Field Descriptions

Bit	Name	Value	Description
31–0	SSW_CAPTURE_COUNT	0–FFFF FFFF	Capture count. This register returns the value of the capture count.

4.2.12 SSW PLL BIST Control Register3 (SSWPLL3)

This is observation register used to log counter value for CLKOUT counter inside PLL wrapper. The SSWPLL3 register is shown in Figure 4-13 and described in Table 4-13. This register is for TI internal use only.

Figure 4-13. SSW PLL BIST Control Register1 (SSWPLL3) [offset = 0x2C]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SSW_CLKOUT_COUNT															
R-0															

R = Read; W = Write; -n = value after reset

Table 4-13. SYS Pin Control Register9 (SYSPC9) Field Descriptions

Bit	Name	Value	Description
31–0	SSW_CAPTURE _COUNT	0–FFFF FFFFh	Value of CLKout count register.

4.2.13 Clock Source Disable Register (CSDIS)

This register is shown in Figure 4-14 and described in Table 4-14.

Figure 4-14. Clock Source Disable Register (CSDIS) [offset = 0x30]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CLKSR7 OFF	CLKSR6 OFF	CLKSR5 OFF	CLKSR4 OFF	CLKSR3 OFF	CLKSR2 OFF	CLKSR1 OFF	CLKSR0 OFF
R-0								R/WP-D	R/WP-D	R/WP-D	R/WP-D	R/WP-D	R/WP-D	R/WP-1	R/WP-0

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset; D = Device-specific reset value

Table 4-14. Clock Source Disable Register (CSDIS) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return zero and writes have no effect.
7–0	CLKSR[7:0] OFF		Clock source[7:0] off.
		0	Clock source[7:0] is enabled.
		1	Clock source[7:0] is disabled.
			Note: On wakeup, only the clock sources that are selected by the device clock domains are awakened (auto wake-up using the wakeup signal).
			Table 4-15 presents the standard mapping for the TMS470M Series clock source.

Table 4-15. Standard Mapping for TMS470M Series Clock Source

Clock Source	Mapping
Clock source0	Oscillator
Clock source1	PLL
Clock source2	32 kHz oscillator
Clock source3	External clock
Clock source4	LPO low frequency clock
Clock source5	LPO high frequency clock
Clock source6 to Clock source7	Reserved for future use

4.2.14 Clock Source Disable Set Register (CSDISSET)

This register is shown in Figure 4-15 and described in Table 4-15.

Figure 4-15. Clock Source Disable Set Register (CSDISSET) [offset = 0x34]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								SETCLKSR7OFF	SETCLKSR6OFF	SETCLKSR5OFF	SETCLKSR4OFF	SETCLKSR3OFF	SETCLKSR2OFF	SETCLKSR1OFF	SETCLKSR0OFF
R-0								R/WP-D	R/WP-D	R/WPD	R/WP-D	R/WP-D	R/WP-D	R/WP-1	R/WP-0

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 4-16. Clock Source Disable Set Register (CSDISSET) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return zero and writes have no effect.
7–0	SETCLKSR[7:0] OFF		Set clock source[7:0] off.
		0	Read: Clock source[7:0] is enabled. Write: Clock source[7:0] is unchanged.
		1	Read: Clock source[7:0] is disabled. Write: Clock source[7:0] is disabled.

4.2.15 Clock Source Disable Clear Register (CSDISCLR)

This register is shown in Figure 4-16 and described in Table 4-16.

Figure 4-16. Clock Source Disable Clear Register (CSDISCLR) [offset = 0x38]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CLRCLKSR7 OFF	CLRCLKSR6 OFF	CLRCLKSR5 OFF	CLRCLKSR4 OFF	CLRCLKSR3 OFF	CLRCLKSR2 OFF	CLRCLKSR1 OFF	CLRCLKSR0 OFF
R-0								R/WP-D	R/WP-D	R/WP-D	R/WP-D	R/WP-D	R/WP-D	R/WP-1	R/WP-0

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 4-17. Clock Source Disable Clear Register (CSDISCLR) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return zero and writes have no effect.
7–0	CLRCLKSR[7:0] OFF		Clear clock source[7:0] off.
		0	Read: Clock source[7:0] is enabled. Write: Clock source[7:0] is unchanged.
		1	Read: Clock source[7:0] is disabled. Write: Clock source[7:0] is enabled.

4.2.16 Clock Domain Disable Register (CDDIS)

This register is shown in Figure 4-17 and described in Table 4-18.

Note: All the clock domains are enabled on wakeup.
The system should guarantee that when HCLK and VCLKSYS are turned off through the HCLKOFF bit, the GCLK domain is also turned off.

Figure 4-17. Clock Domain Disable Register (CSDDIS) [offset = 0x3C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved								RTICLK2OFF	RTICLK1OFF	VCLKA2OFF	VCLKA1OFF	VCLK2OFF	VCLKPOFF	HCLKOFF	GCLKOFF
R-0								R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

Table 4-18. Clock Domain Disable Register (CSDDIS) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return zero and writes have no effect.
7–6	RTICLK[2:1]OFF		RTICLK[2:1] domain off.
		0	The RTICLK[2:1] domain is enabled.
		1	The RTICLK[2:1] domain is disabled.
5–4	VCLKA[2:1]OFF		VCLKA[2:1] domain off.
		0	The VCLKA[2:1] domain is enabled.
		1	The VCLKA[2:1] domain is disabled.
3	VCLK2OFF		VCLK2 domain off.
		0	The VCLK2 domain is enabled.
		1	The VCLK2 domain is disabled.
1	HCLKOFF		HCLK domain off.
		0	The HCLK domain is enabled.
		1	The HCLK domain is disabled.
0	GCLKOFF		GCLK domain off.
		0	The GCLK domain is enabled.
		1	The GCLK domain is disabled.

4.2.17 Clock Domain Disable Set Register (CDDISSET)

This register is shown in Figure 4-18 and described in Table 4-19.

Figure 4-18. Clock Domain Disable Set Register (CSDDISSET) [offset = 0x40]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								SETRTICLK2OFF	SETRTICLK1OFF	SETVCLKA2OFF	SETVCLKAOFF	SETVCLK2OFF	SETVCLKPOFF	SETHCLKOFF	SETGCLKOFF
R-0								R/WP-1	R/WP-1	R/WP-1	R/WP-0	R/WP-1	R/WP-1	R/WP-1	R/WP-0

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset = Device Specific

Table 4-19. Clock Domain Disable Set Register (CSDDISSET) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return zero and writes have no effect.
7–6	SETRTICLK[2:1] OFF		Set RTICLK[2:1] domain.
		0	Read: The RTICLK[2:1] domain is enabled. Write: The RTICLK[2:1] domain is unchanged.
		1	Read: The RTICLK[2:1] domain is disabled. Write: The RTICLK[2:1] domain is disabled.
5	SETVCLKA2OFF		Set VCLKA2 domain.
		0	Read: The VCLKA2 domain is enabled. Write: The VCLK2 domain is unchanged.
		1	Read: The VCLKA2 domain is disabled. Write: The VCLKA2 domain is disabled.
4	SETVCLKAOFF		Set VCLKA domain.
		0	Read: The VCLKA domain is enabled. Write: The VCLKA domain is unchanged.
		1	Read: The VCLKA domain is disabled. Write: The VCLKA domain is disabled.
3	SETVCLK2OFF		Set VCLK2 domain.
		0	Read: The VCLK2 domain is enabled. Write: The VCLK2 domain is unchanged.
		1	Read: The VCLKA domain is disabled. Write: The VCLK2 domain is disabled.

Table 4-19. Clock Domain Disable Set Register (CSDDISSET) Field Descriptions (Continued)

Bit	Name	Value	Description
2	SETVCLKPOFF		Set VCLKP domain.
		0	Read: The VCLKP domain is enabled. Write: The VCLKP domain is unchanged.
		1	Read: The VCLKP domain is disabled. Write: The VCLKP domain is disabled.
1	SETHCLKOFF		Set HCLK domain.
		0	Read: The HCLK1 domain is enabled. Write: The HCLK1 domain is unchanged.
		1	Read: The HCLK1 domain is disabled. Write: The HCLK1 domain is disabled.
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 disabled.

4.2.18 Clock Domain Disable Clear Register (CDDISCLR)

This register is shown in Figure 4-19 and described in Table 4-20.

Figure 4-19. Clock Domain Disable Clear Register (CSDDISCLR) [offset = 0x44]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved								CLR RTI CLK2 OFF	CLR RTI CLK1 OFF	CLR VCLKA 2 OFF	CLR VCLKA OFF	CLR VCLK2 OFF	CLR VCLKP OFF	CLR HCLK OFF	CLR GCLK OFF
R-0								R/WP-1	R/WP-1	R/WP-1	R/WP-0	R/WP-1	R/WP-1	R/WP-1	R/WP-0

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset; D = Device Specific

Table 4-20. Clock Domain Disable Clear Register (CSDDISCLR) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return zero and writes have no effect.
7–6	CLRRTICLK[2:1] OFF		Clear RTICLK[2:1] domain.
		0	Read: The RTICLK[2:1] domain is enabled. Write: The RTICLK[2:1] domain is unchanged.
		1	Read: The RTICLK[2:1] domain is disabled. Write: The RTICLK[2:1] domain is enabled.
5	CLRVCLKA2 OFF		Clear VCLKA2 domain.
		0	Read: The VCLKA2 domain is enabled. Write: The VCLKA2 domain is unchanged.
		1	Read: The VCLKA2 domain is disabled. Write: The VLCKA2 domain is enabled.
4	CLRVCLKAOFF		Clear VCLKA domain.
		0	Read: The VCLKA domain is enabled. Write: The VCLKA domain is unchanged.
		1	Read: The VCLKA domain is disabled. Write: The VCLKA domain is enabled.
3	CLRVCLK2OFF		Clear VCLK2 domain.
		0	Read: The VCLK2 domain is enabled. Write: The VCLK2domain is unchanged.
		1	Read: The VCLK2 domain is disabled. Write: The VCLK2 domain is enabled.

Table 4-20. Clock Domain Disable Clear Register (CSDDISCLR) Field Descriptions (Continued)

Bit	Name	Value	Description
2	CLRVCLKPOFF		Clear VCLKP domain.
		0	Read: The VCLKP domain is enabled. Write: The VCLKPdomain is unchanged.
		1	Read: The VCLKP domain is disabled. Write: The VCLKP domain is enabled.
1	CLRHCLKOFF		Clear HCLK domain.
		0	Read: The HCLK domain is enabled. Write: The HCLK domain is unchanged.
		1	Read: The HCLK domain is disabled. Write: The HCLK domain is enabled.
0	CLRGCLKOFF		Clear GCLK enable.
		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 enabled.

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

This register is shown in Figure 4-20 and described in Table 4-21.

Figure 4-20. GCLK, HCLK, VCLK, and VCLK2 Source Register (GHVSRC) [offset = 0x48]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				GHVWAKE(3:0)				Reserved				HVLPM(3:0)			
R-0				R/WP-0000				R-0				R/WP-0000			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												GHVSRC(3:0)			
R-0												R/WP-0000			

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 4-21. GCLK, HCLK, VCLK, and VCLK2 Source Register (GHVSRC) Field Descriptions

Bit	Name	Value	Description
31–28	Reserved		Reads return zero and writes have no effect.
27–24	GHVWAKE[3:0]		GCLK, HCLK, VCLK, VCLK2 source on wakeup.
		0000	Clock source0 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup.
		0001	Clock source1 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup.
		0010	Clock source2 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup.
		0011	Clock source3 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup.
		0100	Clock source4 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup.
		0101	Clock source5 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup.
		0110	Clock source6 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup.
		0111	Clock source7 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup.
		1000–1111	Reserved
23–20	Reserved		Reads return zero and writes have no effect.
19–16	HVLPM[3:0]		HCLK, VCLK, VCLK2 source on wakeup when GCLK is turned off.
		0000	Clock source0 is the source for HCLK, VCLK, VCLK2 on wakeup.
		0001	Clock source1 is the source for HCLK, VCLK, VCLK2 on wakeup.
		0010	Clock source2 is the source for HCLK, VCLK, VCLK2 on wakeup.
		0011	Clock source3 is the source for HCLK, VCLK, VCLK2 on wakeup.
		0100	Clock source4 is the source for HCLK, VCLK, VCLK2 on wakeup.
		0101	Clock source5 is the source for HCLK, VCLK, VCLK2 on wakeup.

Table 4-21. GCLK, HCLK, VCLK, and VCLK2 Source Register (GHVSRC) Field Descriptions (Continued)

Bit	Name	Value	Description
		0110	Clock source6 is the source for HCLK, VCLK, VCLK2 on wakeup.
		0111	Clock source7 is the source for HCLK, VCLK, VCLK2 on wakeup.
		1000–1111	Reserved
15–4	Reserved		Reads return zero and writes have no effect.
3–0	GHVSRC[3:0]		GCLK, HCLK, VCLK, VCLK2 current source. Note: The GHVSRC[3:0] bits are updated with the HVLPM[3:0] when GCLK is turned off, and are updated with GHVWAKE[3:0] on system wakeup.
		0000	Clock source0 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup.
		0001	Clock source1 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup.
		0010	Clock source2 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup.
		0011	Clock source3 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup.
		0100	Clock source4 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup.
		0101	Clock source5 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup.
		0110	Clock source6 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup.
		0111	Clock source7 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup.
		1000–1111	Reserved

4.2.20 Peripheral Asynchronous Clock Source Register (VCLKASRC)

This register is shown in Figure 4-21 and described in Table 4-22.

Figure 4-21. Peripheral Asynchronous Clock Source Register (VCLKASRC) [offset = 0x4C]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				VCLKA2S(3:0)				Reserved				VCLKA1S(3:0)			
R-0				R/WP-1001				R-0				R/WP-1001			

R = Read in all modes; WP = Write in privileged mode only; -n values after reset

Table 4-22. Peripheral Asynchronous Clock Source Register (VCLKASRC) Field Descriptions

Bit	Name	Value	Description
31–12	Reserved		Reads return zero and writes have no effect.
11–8	VCLKA2S[3:0]		Peripheral asynchronous clock2 source.
		0000	Clock source0 is the source for peripheral asynchronous clock2.
		0001	Clock source1 is the source for peripheral asynchronous clock2.
		0010	Clock source2 is the source for peripheral asynchronous clock2.
		0011	Clock source3 is the source for peripheral asynchronous clock2.
		0100	Clock source4 is the source for peripheral asynchronous clock2.
		0101	Clock source5 is the source for peripheral asynchronous clock2.
		0110	Clock source6 is the source for peripheral asynchronous clock2.
		0111	Clock source7 is the source for peripheral asynchronous clock2.
		1000–1111	VCLK is the source for peripheral asynchronous clock2.
7–4	Reserved		Reads return zero and writes have no effect.

Table 4-22. Peripheral Asynchronous Clock Source Register (VCLKASRC) Field Descriptions (Continued)

Bit	Name	Value	Description
3–0	VCLKA1S[3:0]		Peripheral asynchronous clock1 source.
		0000	Clock source0 is the source for peripheral asynchronous clock1.
		0001	Clock source1 is the source for peripheral asynchronous clock1.
		0010	Clock source2 is the source for peripheral asynchronous clock1.
		0011	Clock source3 is the source for peripheral asynchronous clock1.
		0100	Clock source4 is the source for peripheral asynchronous clock1.
		0101	Clock source5 is the source for peripheral asynchronous clock1.
		0110	Clock source6 is the source for peripheral asynchronous clock1.
		0111	Clock source7 is the source for peripheral asynchronous clock1.
		1000–1111	VCLK is the source for peripheral asynchronous clock1.

4.2.21 RTI Clock Source Register (RCLKSRC)

This register is shown in Figure 4-22 and described in Table 4-23.

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 4-22. RTI Clock Source Register (RCLKSRC) [offset = 0x50]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						RTI2DIV(1:0)		Reserved				RTI2SRC(3:0)			
R-0						R/WP-01		R-0				R/WP-1001			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						RTI1DIV(1:0)		Reserved				RTI1SRC(3:0)			
R-0						R/WP-01		R-0				R/WP-1001			

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

Table 4-23. RTI Clock Source Register (RCLKSRC) Field Descriptions

Bit	Name	Value	Description
31–26	Reserved		Reads return zero and writes have no effect.
25–24	RTI2DIV[1:0]		RTI clock2 Divider.
		00	Prescaler divide value of RTICLK2 source is 1.
		01	Prescaler divide value of RTICLK2 source is 2.
		10	Prescaler divide value of RTICLK2 source is 4.
		11	Prescaler divide value of RTICLK2 source is 8.
23–20	Reserved		Reads return zero and writes have no effect.
19–16	RTI2SRC[3:0]		RTI clock2 source.
		0000	Clock source0 is the source for RTICLK2.
		0001	Clock source1 is the source for RTICLK2.
		0010	Clock source2 is the source for RTICLK2.
		0011	Clock source3 is the source for RTICLK2.
		0100	Clock source4 is the source for RTICLK2.
		0101	Clock source5 is the source for RTICLK2.
		0110	Clock source6 is the source for RTICLK2.
		0111	Clock source7 is the source for RTICLK2.

Table 4-23. RTI Clock Source Register (RCLKSRC) Field Descriptions (Continued)

Bit	Name	Value	Description
		1000–1111	VCLK is the source for RTICLK2.
15–10	Reserved		Reads return zero and writes have no effect.
9–8	RTI1DIV[1:0]		RTI clock1 divider.
		00	Prescaler divide value of RTICLK1 source is 1.
		01	Prescaler divide value of RTICLK1 source is 2.
		10	Prescaler divide value of RTICLK1 source is 4.
		11	Prescaler divide value of RTICLK1 source is 8.
7–4	Reserved		Reads return zero and writes have no effect.
3–0	RTI1SRC[3:0]		RTI clock1 source.
		0000	Clock source0 is the source for RTICLK1.
		0001	Clock source1 is the source for RTICLK1.
		0010	Clock source2 is the source for RTICLK1.
		0011	Clock source3 is the source for RTICLK1.
		0100	Clock source4 is the source for RTICLK1.
		0101	Clock source5 is the source for RTICLK1.
		0110	Clock source6 is the source for RTICLK1.
		0111	Clock source7 is the source for RTICLK1.
		1000–1111	VCLK is the source for RTICLK1.

4.2.22 Clock Source Valid Status Register (CSVSTAT)

This register is shown in Figure 4-23 and described in Table 4-24.

Figure 4-23. Clock Source Valid Status Register (CSVSTAT) [offset = 0x54]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CLKSR7V	CLKSR6V	CLKSR5V	CLKSR4V	CLKSR3V	CLKSR2V	CLKSR1V	CLKSR0V
R-0								R-D	R-D	R-D	R-D	R-D	R-D	R-D	R-D

R = Read all modes; -n = Value after power-up reset; D = Device Specific

.

Table 4-24. Clock Source Valid Register (CSVSTAT) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return zero and writes have no effect.
7–0	CLKSR[7:0]V		Clock source[7:0] valid.
		0	Clock source[7:0] is not valid.
		1	Clock source[7:0] is valid.
			Note: If the valid bit of the source of a clock domain is not set (i.e., the clock source is not fully stable), the respective clock domain is disabled by the GCM.

4.2.23 Memory Self-Test Global Control Register (MSTGCR)

This register is shown in Figure 4-24 and described in Table 4-25.

Figure 4-24. Memory Self-Test Global Control Register (MSTGCR) [offset = 0x58]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								MBIST_ALGSEL(7:0)							
R-0								R/WP-00000000							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						ROM_DIV(1:0)		Reserved				MSTGENA(3:0)			
R-0						R/WP-00		R-0				R/WP-0101			

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 4-25. Memory Self-Test Global Control Register (MSTGCR) Field Descriptions

Bit	Name	Value	Description
31–24	Reserved		Reads return zero and writes have no effect.
23-16	MBIST_ALGSEL		Selects different Algorithm for MBIST
		00000000	No Algorithm selected.
		00000001	Checkerboard has been selected.
		00000010	March 13N with background of all 0s and all 1s has been selected.
		00000100	March 11N with background of hex 5 and As has been selected.
		00001000	March 13N with backgrounds of hex 3 and Cs, hex 0F and F0s, and 69 and 96s has been selected.
		00010000	PMOS Open Address Decode has been selected.
		00100000	No Algorithm selected.
		01000000	No Algorithm selected.
		10000000	No Algorithm selected.
15-10	Reserved		Reads return zero and writes have no effect.

Table 4-25. Memory Self-Test Global Control Register (MSTGCR) Field Descriptions (Continued)

Bit	Name	Value	Description
9–8	ROM_DIV[1:0]		Prescaler divider bits for ROM clock source.
		00	ROM clock source is HCLK divided by 1. MBIST will reset for 16 VBUS cycles.
		01	ROM clock source is HCLK divided by 2. MBIST will reset for 32 VBUS cycles.
		10	ROM clock source is HCLK divided by 4. MBIST will reset for 64 VBUS cycles.
		11	ROM clock source is HCLK divided by 8. MBIST will reset for 96 VBUS cycles.
7–4	Reserved		Reads return zero and writes have no effect.
3–0	MSTGENA[3:0]		Memory self-test controller global enable key Note: Enabling the MSTGENA key will generate a reset (MSTC_RST_L) to the state machines of all the selected MBIST controllers. It also selects the configuration of all the MBIST controllers in the device to be done by the memory self-test controller instead of the test interface. Note: It is recommended to write a value of 0101 to disable the MSTGCENA key to avoid enabling by soft error.
		1010	Memory self-test controller is enabled.
		Others	Memory self-test controller is disabled.

4.2.24 Memory Hardware Initialization Global Control Register (MINITGCR)

This register is shown in Figure 4-25 and described in Table 4-26.

Figure 4-25. Memory Hardware Initialization Global Control Register (MINITGCR) [offset = 0x5C]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												MINITGENA(3:0)			
R-0												R/WP-0101			

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 4-26. Memory Hardware Initialization Global Control Register (MINITGCR) Field Descriptions

Bit	Name	Value	Description
31–4	Reserved		Reads return zero and writes have no effect.
3–0	MINITGENA[3:0]		Memory hardware initialization global enable key. Note: It is recommended to write a value of 0101 to disable the MINITGENA key to avoid enabling by soft error.
		1010	Global memory hardware initialization key is enabled
		Others	Global memory hardware initialization key is disabled

4.2.25 MBIST Controller/ Memory Initialization Enable Register (MSINENA)

This register is shown in Figure 4-26 and described in Table 4-27.

Figure 4-26. MBIST Controller/Memory Initialization Enable Register (MSINENA) [offset = 0x60]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSIENA[31:16]															
R/WP-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSIENA[15:0]															
R/WP-0															

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 4-27. MBIST Controller/Memory Initialization Enable Register (MSINENA) Field Descriptions

Bit	Name	Value	Description
31–0	MSIENA[31:0]		MBIST controller/memory initialization enable register. 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. In memory self-test mode, all the selected MSIENA[31:0] bits need to be enabled before enabling the global memory self-test controller key (MSTGENA). The reason for this is that MSTGENA, in addition to being the global enable for all individual MBIST controllers is the source for the reset generation to all the MBIST controller state machines. It also selects the controller inputs to the MBIST controllers from the memory self-test controller (MSTC) instead of the test interface Disabling the MSTGENA or MINITGENA key (by writing from a 1010 to any other value) will reset all the MSIENA[31:0] bits to their default values.
		0	In memory self-test mode (MSTGENA = 1010): MBIST controller [31:0] is disabled. In memory Initialization mode (MINITGNA = 1010): Memory module [31:0] hardware initialization is disabled.
		1	In memory self-test mode (MSTGENA = 1010): MBIST controller [31:0] is enabled. In memory Initialization mode (MINITGNA = 1010): Memory module[31:0] hardware initialization is enabled.

4.2.26 Memory Self-Test Fail Status Register (MSTFAIL)

This register is shown in Figure 4-27 and described in Table 4-28.

Figure 4-27. Memory Self-Test Fail Status Register (MSTFAIL) [offset = 0x64]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSTF[31:16]															
R/WP-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MSTF[15:0]															
R/WP-0															

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

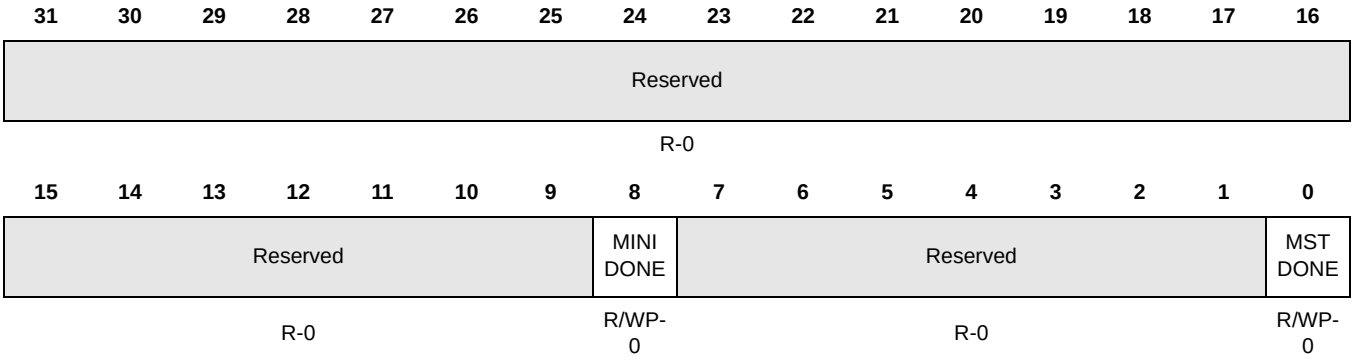
Table 4-28. Memory Self-Test Fail Status Register (MSTFAIL) Field Descriptions

Bit	Name	Value	Description
31–0	MSTF[31:0]		Memory self-test fail status bit. Note: Disabling the MSTGENA key (by writing from 1010b to any other value) will reset all the individual fail status bits to their default values.
		0	Read: MBIST controller [31:0] run did not fail. Write: A write of 0 has no effect.
		1	Read: MBIST controller [31:0] run failed. Write: The bit is cleared to 0.

4.2.27 MSTC Global Status Register (MSTCGSTAT)

This register is shown in Figure 4-28 and described in Table 4-29.

Figure 4-28. MSTC Global Status Register (MSTCGSTAT) [offset = 0x68]



R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

Table 4-29. MSTC Global Status Register (MSTCGSTAT) Field Descriptions

Bit	Name	Value	Description
31–9	Reserved		Reads return zero and writes have no effect.
8	MINIDONE		Memory hardware initialization test run complete status.
		0	Read: Memory hardware initialization is not completed. Write: A write of 0 has no effect.
		1	Read: Memory hardware initialization is completed. Write: A write of 1 has no effect.
			Note: In order to clear MINIDONE bit field, MINITGENA key in register MINITGCR (0x5C) needs to be disable by writing from 1010 to any other value preferably 0101.
7–1	Reserved		Reads return zero and writes have no effect.
0	MSTDONE		Memory self-test complete status. Note: Disabling the MSTGENA key (by writing from a 1010 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.

4.2.28 Memory Hardware Initialization Status Register (MINISTAT)

This register is shown in Figure 4-29 and described in Table 4-30.

Figure 4-29. Memory Hardware Initialization Status Register (MINISTAT) [offset = 0x6C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MIDONE[31:16]															
R/WP-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MIDONE[15:0]															
R/WP-0															

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 4-30. Memory Hardware Initialization Status Register (MINISTAT) Field Descriptions

Bit	Name	Value	Description
31–0	MIDONE[31:0]		Memory hardware initialization status bit. Note: Disabling the MINITGENA key (by writing from a 1010 to any other value) will reset all the individual fail status bits to 0.
		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.

4.2.29 PLL Control Register 1 (PLLCTL1)

This register is device specific and a detailed description can be found in section 6.4.1.1.

This register is shown in Figure 6-8 and described in Table 6-2.

4.2.30 PLL Control Register 2 (PLLCTL2)

This register is device specific and a detailed description can be found in section 6.4.1.2.

This register is shown in Figure 6-9 and described in Table 6-3.

4.2.31 Die Identification Register Lower Word (DIEIDL)

This register is shown in Figure 4-30 and described in Table 4-31.

Figure 4-30. Die Identification Register, Lower Word (DIEIDL) [offset = 0x7C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DIEIDL															
R-X															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DIEIDL															
R-X															

R = Read only; -X = Value unchanged after reset

.

Table 4-31. Die Identification Register, Lower Word (DIEIDL) Field Descriptions

Bit	Name	Value	Description
31–0	DIEIDL(16–0)	0–FFFF_FFFFh	This read-only register contains the lower word (31:0) of the die ID information.

4.2.32 Die Identification Register Upper Word (DIEIDH)

This register is shown in Figure 4-31 and described in Table 4-32.

Figure 4-31. Die Identification Register, Upper Word (DIEIDH) [offset = 0x80]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DIEIDH															
R-X															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DIEIDH															
R-X															

R = Read only; -X = Value unchanged after reset

.

Table 4-32. Die Identification Register, Upper Word (DIEIDH) Field Descriptions

Bit	Name	Value	Description
31–0	DIEIDH(63–32)	0–FFFF_FFFFh	This read-only register contains the upper word (63–32) of the die ID information.

4.2.33 Voltage Regulator Control Register (VRCTL)

This register is shown in Figure 4-32 and described in Table 4-33.

Figure 4-32. Voltage Regulator Control Register (VRCTL) [offset = 0x84]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								VLPMENA				VSLEEPENA			
R-0								R/WP-0				R/WP-0			

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 4-33. Voltage Regulator Control Register (VRCTL) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return zero and writes have no effect.
7–4	VLPMENA[3:0]		Voltage regulator global low power modes enable.
		0000–1110	The voltage regulator is in normal mode.
		1111	The voltage regulator is in sleep mode when GCLK, HCLK/VCLK-SYS, and VCLKP are gated and the voltage regulator global low power mode is enabled (VLPMENA = 1111).
3–0	VSLEEPENA[3:0]		Voltage regulator sleep mode enable. Table 4-34 describes the system configuration to control the voltage regulator's mode.
		0000–1110	The voltage regulator is in non-sleep mode.
		1111	The voltage regulator is in halt mode (VSLEEPENA = 0000-1110) or sleep mode (VSLEEPENA = 1111) when GCLK, HCLK/VCLKSYS, and VCLKP are gated.

Table 4-34. System Configuration to Control Voltage Regulator Modes and Wakeup Action

VLP MENA	VSLEEP ENA	GCLK, HCLK, VBUSP Clocks OFF	MODE 1	MODE 2	Voltage Regulator Mode	Wakeup Action	Wakeup Source to Enable Clocks
0000–1110	X	Yes	0	0	Normal mode	System wakeup enables all the clock domains.	System wakeup
1111	0000–1110	Yes	0	1	Halt mode	On system wakeup, the voltage regulator is configured to be in normal mode and the NORMRRDY (normal mode ready) of the voltage regulator in turn wakes up the disabled clocks	Normal mode ready (NORMRRDY) of the voltage regulator and with the system wakeup
1111	1111	Yes	1	1	Sleep mode	Dedicated voltage regulator wakes up pins. Wake up the voltage regulator from sleep mode to normal mode.	System reset caused by wakeup from sleep mode to normal mode
X	X	No	0	0	Normal mode	System wakeup enables all the clock domains.	System wakeup

4.2.34 LPO/Clock Monitor Control Register (LPOMONCTL)

This register is shown in Figure 4-33 and described in Table 4-35.

Figure 4-33. LPO/Clock Monitor Control Register (LPOMONCTL) [offset = 0x88]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved							BIAS ENABLE	Reserved							
R-0							R/WP-1	R-0							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				HFTRIM(3:0)				Reserved				LFTRIM(3:0)			
R-0				R/WP-1000				R-0				R/WP-1000			

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

Table 4-35. LPO/Clock Monitor Control Register (LPOMONCTL) Field Descriptions

Bit	Name	Value	Description
31–25	Reserved		Reads return zero and 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.
23–12	Reserved		Reads return zero and writes have no effect.
11–8	HFTRIM[3:0]		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 of SYSCLK, otherwise a system failure could result. The values below are the ratio: f / fo
		0000	50.00%
		0001	56.25%
		0010	62.50%
		0011	68.75%
		0111	75.00%
		0100	81.25%
		0101	87.50%
		0110	100.00%
		1000	106.25%
		1001	112.50%

Table 4-35. LPO/Clock Monitor Control Register (LPOMONCTL) Field Descriptions (Continued)

Bit	Name	Value	Description
		1010	118.75%
		1011	125.00%
		1100	131.25%
		1101	137.50%
		1110	143.75%
		1111	150.00%
7–4	Reserved		Reads return zero and writes have no effect.
3–0	LFTRIM[3:0]		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 of SYSCLK, otherwise a crash could result. The values below are the ratio: f / fo
		0000	50.00%
		0001	56.25%
		0010	62.50%
		0011	68.75%
		0111	75.00%
		0100	81.25%
		0101	87.50%
		0110	100.00%
		1000	106.25%
		1001	112.50%
		1010	118.75%
		1011	125.00%
		1100	131.25%
		1101	137.50%
		1110	143.75%
		1111	150.00%

4.2.35 Clock Test Register (CLKTEST)

This register is shown in Figure 4-34 and described in Table 4-36.

Figure 4-34. Clock Test Register (CLKTEST) [offset = 0x8C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						RANGE DET CTRL	RANGE DET ENA SSEL	Reserved				CLK_TEST_EN(3:0)			
R-0						R/WP-0	R/WP-0	R-0				R/WP-1010			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				SEL_GIO_PIN(3:0)				Reserved				SEL_ECP_PIN(3:0)			
R-0				R/WP-0000				R-0				R/WP-0000			

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 4-36. Clock Test Register (CLKTEST) Field Descriptions

Bit	Name	Value	Description
31–26	Reserved		Reads return zero and writes have no effect.
25	RANGEDETCTRL		Range detection control.
		0	The clock monitor range detection circuitry (RANGEDETECTENABLE) is disabled.
		1	The clock monitor range detection circuitry (RANGEDETECTENABLE) is enable.d
24	RANGEDETENSSEL		Selects range detection enable. This bit resets asynchronously on SYS_nRST.
		0	The message range detect enable is generated by the hardware in the clock monitor wrapper.
		1	The message range detect enable is controlled by the bit RANGEDETCTRL (CLKTEST[25]).
20–16	CLK_TEST_EN[3:0]		Clock test enable. This bit field enables or disables clock going to device pins. Two pins in a device can get clock sources by enabling CLK_TEST_EN bits. One pin is the ECP and second pin is a device specific GIO pin.
		0101	Clocks going to device are enabled.
		Others	Clocks going to device are disabled.
15–12	Reserved		Reads return zero and writes have no effect.

Table 4-36. Clock Test Register (CLKTEST) Field Descriptions (Continued)

Bit	Name	Value	Description
11–8	SEL_GIO_PIN[3:0]		Select the clock source valid signal or clock source at functional GIO pin
		0000	Clock source0 valid signal.
		0001	Clock source1 valid signal.
		0010	Clock source2 valid signal.
		0011	Clock source3 valid signal.
		0100	Clock source4 valid signal.
		0101	Clock source5 valid signal.
		0110	Clock source6 valid signal.
		0111	Clock source7 valid signal.
		1000	Clock source4 is selected on pin.
		1001–1111	Reserved
7–4	Reserved		Reads return zero and writes have no effect.
3–0	SEL_ECP_PIN[3:0]		Select the clock at ECP Pin
		0000	Oscillator clock
		0001	PLL clock
		0010	32 KHz oscillator clock
		0011	External clock
		0100	LPO low frequency clock
		0101	LPO high frequency clock
		0110	Reserved
		0111	Reserved
		1000	GCLKMCLK
		1001	RTICLK1SRC
		1010	RTICLK2SRC
		1011	AVCLK1
		1100	AVCLK2
		1101–1111	Reserved

Note:
The clock test register is for testing purposes. Only valid clock sources can be selected for the output pin.

Details of specific GIO pin getting clock source can be found in the device-specific data sheet. In case of disabling CLK_TEST_EN bits original functionality of ECP pin will be preserved.

Note:
Selecting RTICLK1SRC or RTICLK2SRC as the clock source for ECP brings out VCLK on the ECP pin in normal operation.

4.2.36 General Purpose Register 1(GPREG1)

This register is shown in Figure 4-36 and described in Table 4-38.

Figure 4-35. General Purpose Register (GPREG1) [offset = 0xA0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Device Specific															
R-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Device Specific															
R-0															

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset; D = device specific

.

Table 4-37. General Purpose Register (GPREG1) Field Descriptions

Bit	Name	Value	Description
31–0	Device Specific		Refer to the device datasheet for more details.

4.2.37 BOOT ROM Select Register (BTRMSEL)

This register is shown in Figure 4-36 and described in Table 4-38.

Figure 4-36. BOOT ROM Select Register (BTRMSEL) [offset = 0xA4]

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

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

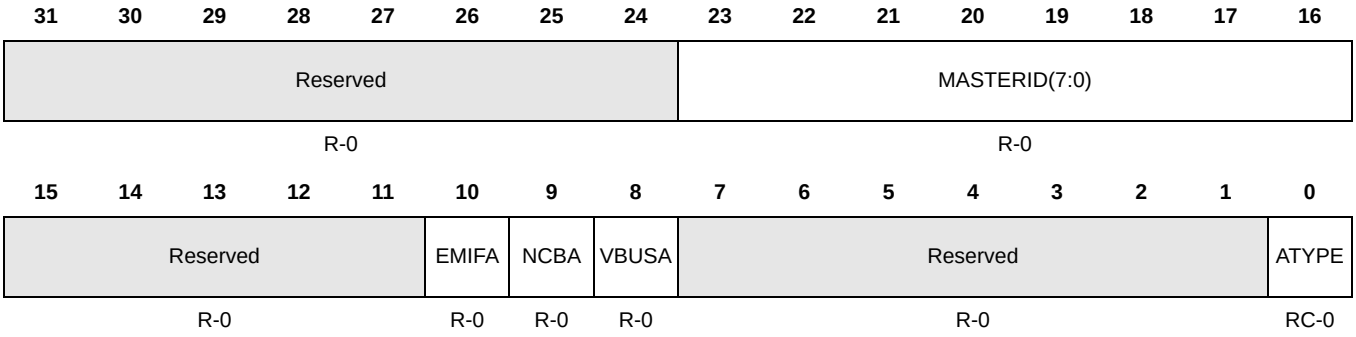
Table 4-38. BOOT ROM Select Register (BTRMSEL) Field Descriptions

Bit	Name	Value	Description
31-0	Reserved		Reads return zero and writes have no effect.

4.2.38 Imprecise Fault Status Register (IMPFASTS)

This register is shown in Figure 4-37 and described in Table 4-39.

Figure 4-37. Imprecise Fault Status Register (IMPFASTS) [offset = 0xA8]



R = Read only; C = Clear by reading; -0 value after power-up reset

.

Table 4-39. Imprecise Fault Status Register (IMPFASTS) Field Descriptions

Bit	Name	Value	Description
31–24	Reserved		Reads return zero and writes have no effect.
23–16	MASTERID[7:0]	0–FFh	Master ID. This register indicates which master is responsible for the imprecise abort. The master ID value depends on device implementation: please refer to the device-specific datasheet for valid MASTERID value. Note: This register is only updated when an imprecise abort occurs. Note: This register is cleared to 00h only on power-up reset. The value of this register remains unchanged after all other resets.
15–11	Reserved		Reads return zero and writes have no effect.
10	EMIFA		EMIF imprecise abort. This register indicates the imprecise abort was generated writing into the EMIF. Note: This register is only updated when an imprecise abort occurs. Note: This register is cleared to 0 only on power-up 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.

Table 4-39. Imprecise Fault Status Register (IMPFASTS) Field Descriptions (Continued)

Bit	Name	Value	Description
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. Note: This register is only updated when an imprecise abort generated by a non-cacheable, bufferable write or shared device write occurs. Note: This register is cleared to 0 only on power-up 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. Note: This register is only updated when an imprecise abort is generated when writing into the peripheral frame. Note: This register is cleared to 0 only on power-up 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		Reads return zero and writes have no effect.
0	ATYPE	0–3Fh	Abort type.This bit indicates to the CPU whether the last abort was an imprecise abort or a precise abort. Notes: 1) This register is updated after each abort is generated to the CPU. 2) This register is cleared on CPU read. 3) This register is cleared to 0 only on power-up reset. The value of this register remains unchanged after all other resets
		0	A precise abort was generated or a new abort was generated. MASTERID, VBUSA, NCBA, EMIFA and IMPFTADD were not updated.
		1	An imprecise abort was generated. MASTERID, VBUSA, NCBA, EMIFA and IMPFTADD were updated. Note: When ATYPE is set, the IMPFAWADD and IMPFASTS bits are frozen, i.e., they are not updated by subsequent ABORT signals.

4.2.39 Imprecise Fault Address Register (IMPFTADD)

This register is shown in Figure 4-38 and described in Table 4-40.

Figure 4-38. Imprecise Fault Write Address Register (IMPFTADD) [offset = 0xAC]

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

R = Read only; -n = value after power-up reset

Table 4-40. Imprecise Fault Write Address R (IMPFTADD) Field Descriptions

Bit	Name	Value	Description
31–0	IMPFTADD[31:0]	0–FFFF FFFF	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 0x0000 0000 only on power-up reset. The value of this register remains unchanged after all other resets.

4.2.40 System Software Interrupt Request 1 Register (SSIR1)

This register is shown in Figure 4-39 and described in Table 4-41.

Figure 4-39. System Software Interrupt Request 1 Register (SSIR1) [offset = 0xB0]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SSKEY1								SSDATA1							
W-0								R/W-0							

R = Read; W = Write in all modes; -n = Value after reset

.

Table 4-41. System Software Interrupt Request 1 Register (SSIR1) Field Descriptions

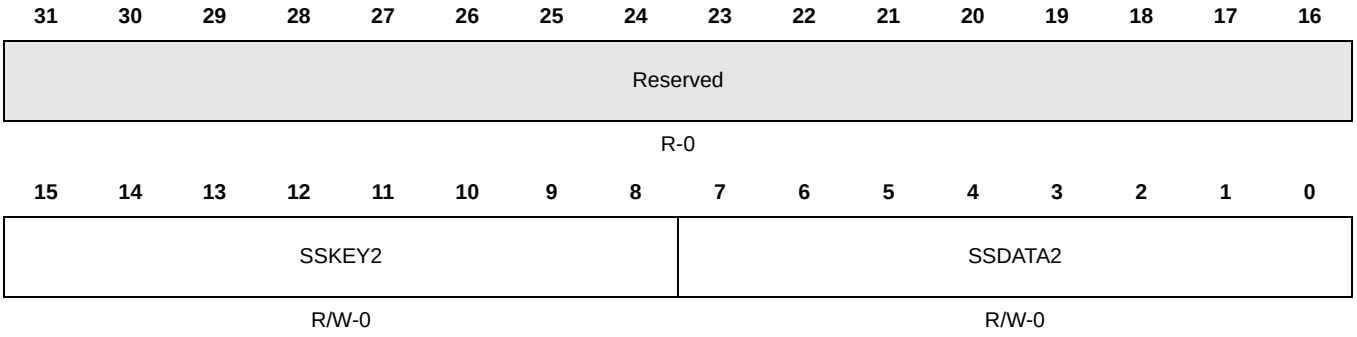
Bit	Name	Value	Description
31–16	Reserved		Reads return zero and writes have no effect.
15–8	SSKEY1[7:0]	0–3Fh	System software interrupt request key. A 75h written to these bits initiates IRQ/FIQ interrupts. Data in this field is always read as 0. The SSKEY1 field can be written into only if the write 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[7:0]	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 9ffset = FCh for compatibility reasons.

4.2.41 System Software Interrupt Request 2 Register (SSIR2)

This register is shown in Figure 4-40 and described in Table 4-42.

Figure 4-40. System Software Interrupt Request 2 Register (SSIR2) [offset = 0xB4]



R = Read; W = Write; -n = Value after reset

.

Table 4-42. System Software Interrupt Request 2 Register (SSIR2) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Reads return zero and writes have no effect.
15–8	SSKEY2[7:0]	84h	System software interrupt2 request key. Read: Data in this field is always read as 0. Write: 84h written to these bits initiates IRQ/FIQ interrupts. The SSKEY2 field can be written into only if the write data matches the key (84h).
7–0	SSDATA2[7:0]	0–FF	System software interrupt2 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.

4.2.42 System Software Interrupt Request 3 Register (SSIR3)

This register is shown in Figure 4-41 and described in Table 4-43.

Figure 4-41. System Software Interrupt Request 3 Register (SSIR3) [offset = 0xB8]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SSKEY3								SSDATA3							
R/W--0								R/W--0							

R = Read; W = Write; U = Undefined; -n = Value after reset

.

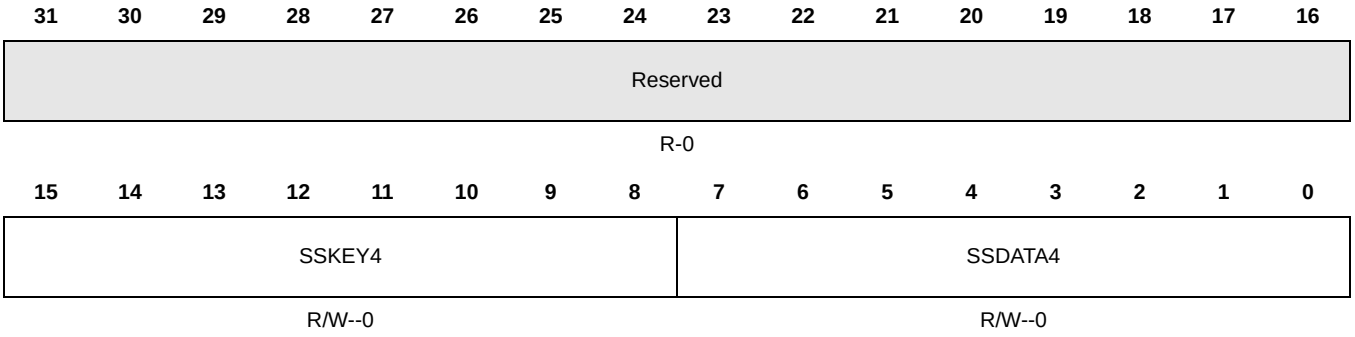
Table 4-43. System Software Interrupt Request 3 Register (SSIR3) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Reads return zero and writes have no effect.
15–8	SSKEY3[7:0]	93h	System software interrupt3 request key. Read: Data in this field is always read as 0. Write: 93h written to these bits initiates IRQ/FIQ interrupts. The SSKEY3 field can be written into only if the write data matches the key (93h).
7–0	SSDATA3[7:0]	0–FF	System software interrupt3 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.

4.2.43 System Software Interrupt Request 4 Register (SSIR4)

This register is shown in Figure 4-42 and described in Table 4-44.

Figure 4-42. System Software Interrupt Request 4 Register (SSIR4) [offset = 0xBC]



R = Read; W = Write; U = Undefined; -n = Value after reset

.

Table 4-44. System Software Interrupt Request 4 Register (SSIR4) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Reads return zero and writes have no effect.
15–8	SSKEY4[7:0]	A2h	System software interrupt3 request key. Read: Data in this field is always read as 0. Write: A2h written to these bits initiates IRQ/FIQ interrupts. The SSKEY4 field can be written into only if the write data matches the key (A2h).
7–0	SSDATA4[7:0]	0–FF	System software interrupt4 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.

4.2.44 RAM Control Register (RAMGCR)

This register is shown in Figure 4-43 and described in Table 4-45.

Figure 4-43. RAM Control Register (RAMGCR) [offset = 0xC0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												RAM_DFT_EN(3:0)			
R-0												R/WP-0101			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	WST_AENA3	Reserved	WST_DENA3	Reserved	WST_AENA2	Reserved	WST_DENA2	Reserved	WST_AENA1	Reserved	WST_DENA1	Reserved	WST AENA0	Reserved	WST DENA0
R-0	R/WP-0	R-0	R/WP-0	R-0	R/WP-0	R-0	R/WP-0	R-0	R/WP-0	R-0	R/WP-0	R-0	R/WP-0	R-0	R/WP-0

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 4-45. RAM Control Register (RAMGCR) Field Descriptions

Bit	Name	Value	Description
31–20	Reserved		Reads return zero and writes have no effect.
19-16	RAM_DFT_EN[3:0]		Functional mode RAM DFT port enable key.
		1010	RAM DFT port is enable.
		Others	RAM DFT port is disabled
15	Reserved		Reads return zero and writes have no effect.
14	WST_AENA3		eSRAM3 address phase wait state enable bit.
		0	The default address setup time for eSRAM3 is used.
		1	The eSRAM3 address setup time is increased by one HCLK cycle.
13	Reserved		Reads return zero and writes have no effect.
12	WST_DENA3		eSRAM3 data phase wait state enable bit.
		0	There are no wait states for eSRAM3 during the data phase.
		1	There is one wait state for eSRAM3 during the data phase.
11	Reserved		Reads return zero and writes have no effect.
10	WST_AENA2		eSRAM2 data phase wait state enable bit.
		0	The default address setup time for eSRAM3 is used.
		1	The eSRAM2 address setup time is increased by one HCLK cycle.
9	Reserved		Reads return zero and writes have no effect.
8	WST_DENA2		eSRAM2 data phase wait state enable bit.

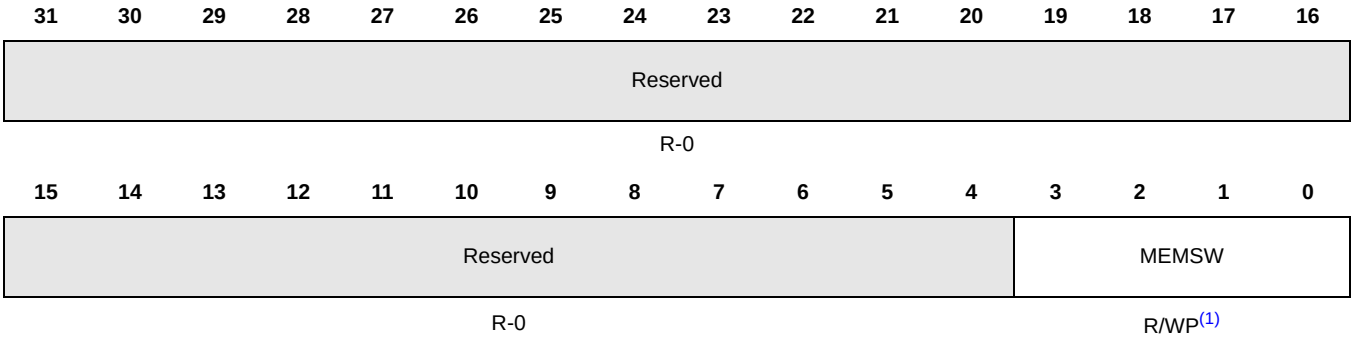
Table 4-45. RAM Control Register (RAMGCR) Field Descriptions (Continued)

Bit	Name	Value	Description
		0	There are no wait states for eSRAM2 during the data phase.
		1	There is one wait state for eSRAM2 during the data phase.
7	Reserved		Reads return zero and writes have no effect.
6	WST_AENA1		eSRAM1 address phase wait state enable bit.
		0	The default address setup time for eSRAM1 is 0.
		1	The eSRAM1 address setup time is increased by one HCLK cycle.
5	Reserved		Reads return zero and writes have no effect.
4	WST_DENA1		eSRAM1 data phase wait state enable bit.
		0	There are no wait states for eSRAM1 during the data phase.
		1	There is one wait state for eSRAM1 during the data phase.
3	Reserved		Reads return zero and writes have no effect.
2	WST_AENA0		eSRAM0 data phase wait state enable bit.
		0	The default address setup time for eSRAM3 is used.
		1	The eSRAM2 address setup time is increased by one HCLK cycle.
1	Reserved		Reads return zero and writes have no effect.
0	WST_DENA0		eSRAM0 data phase wait state enable bit.
		0	There are no wait states for eSRAM0 during the data phase.
		1	There is one wait state for eSRAM0 during the data phase.

4.2.45 Bus Matrix Module Control Register1 (BMMCR1)

This register is shown in Figure 4-44 and described in Table 4-46.

Figure 4-44. Bus Matrix Module Control Register1 (BMMCR) [offset = 0xC4]



R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset
1) The value of MEMSW on system reset is the polarity of the device RAM_MODE pin.

.

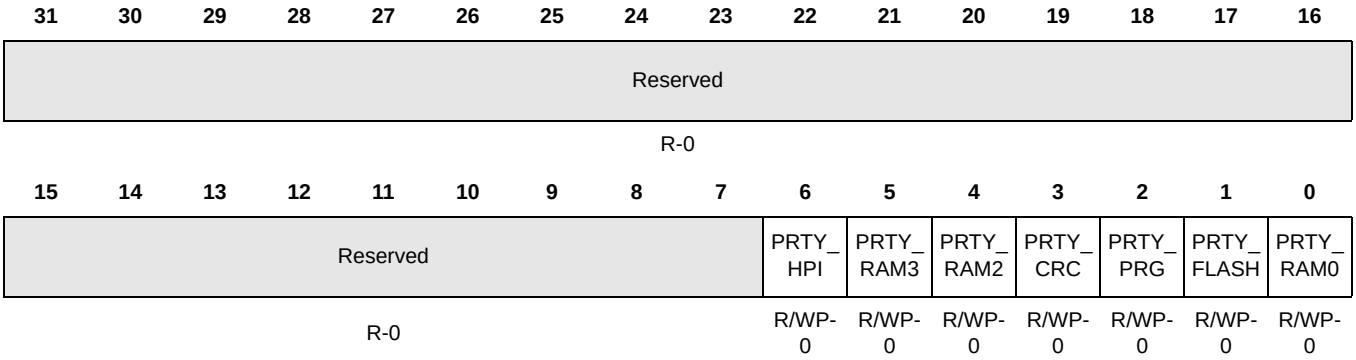
Table 4-46. Bus Matrix Module Control Register1 (BMMCR) Field Descriptions

Bit	Name	Value	Description
31–4	Reserved		Reads return zero and writes have no effect.
3–0	MEMSW[3:0]		Memory swap bit key. Refer to section 1.3 for further explanation on the memory configuration. Notes: 1) The value of MEMSW on system reset is the polarity of the device RAM_MODE pin. 2) Writing any other value than 1010 and 0101 has no effect, and the state of the memory is unchanged.
		1010	Default memory map: Program memory starts at 0x0000_0000 to 0x07FF_FFFF. eSRAM starts at address 0x0800_0000 to 0x0FFF_FFFF.
		0101	Swapped memory map: eSRAM starts at address 0x0000_0000 to 0x07FF_FFFF. Program memory starts at address 0x0800_0000 to 0x0FFF_FFFF

4.2.46 Bus Matrix Module Control Register2 (BMMCR2)

This register is shown in Figure 4-45 and described in Table 4-47.

Figure 4-45. Bus Matrix Module Control Register2 (BMMCR2) [offset = 0xC8]



R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 4-47. Bus Matrix Module Control Register2 (BMMCR2) Field Descriptions

Bit	Name	Value	Description
31–7	Reserved		Reads return zero and writes have no effect.
6	PRTY_HPI		Arbitration priority to HPI.
		0	Fixed priority is used.
		1	Round robin priority is used.
5	PRTY_RAM3		Arbitration priority to eSRAM3.
		0	Fixed priority is used.
		1	Round robin priority is used.
4	PRTY_RAM2		Arbitration priority to eSRAM2.
		0	Fixed priority is used.
		1	Round robin priority is used.
3	PRTY_CRC		Arbitration priority to CRC.
		0	Fixed priority is used.
		1	Round robin priority is used.
2	PRTY_PRG		Arbitration priority to peripheral bridge.
		0	Fixed priority is used.
		1	Round robin priority is used.

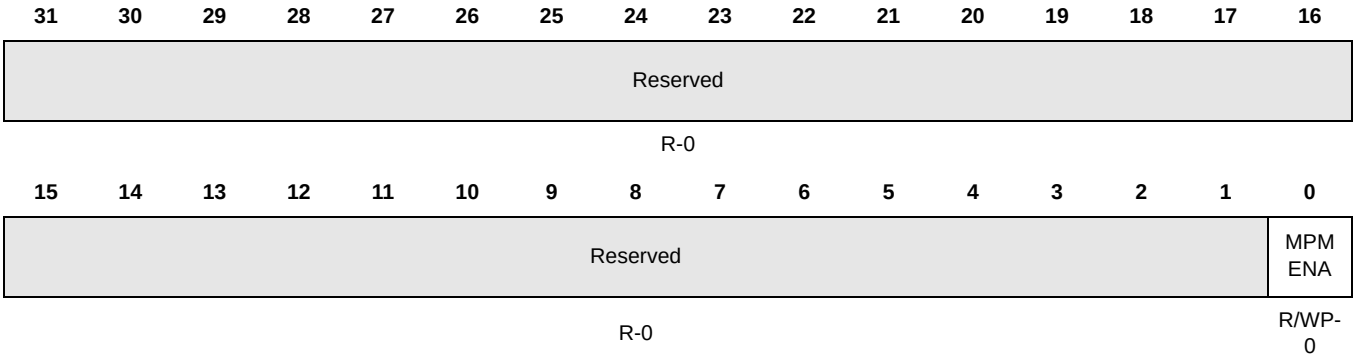
Table 4-47. Bus Matrix Module Control Register2 (BMMCR2) Field Descriptions (Continued)

Bit	Name	Value	Description
1	PRTY_FLASH		Arbitration priority to eSRAM1.
		0	Fixed priority is used.
		1	Round robin priority is used.
0	PRTY_RAM0		Arbitration priority to eSRAM0.
		0	Fixed priority is used.
		1	Round robin priority is used.

4.2.47 MMU Global Control Register (MMUGCR)

This register is shown in Figure 4-46 and described in Table 4-48.

Figure 4-46. MMU Global Control Register (MMUGCR) [offset = 0xCC]



R = Read in all modes; WP = Write in privileged mode only; -n = value after power-up reset

.

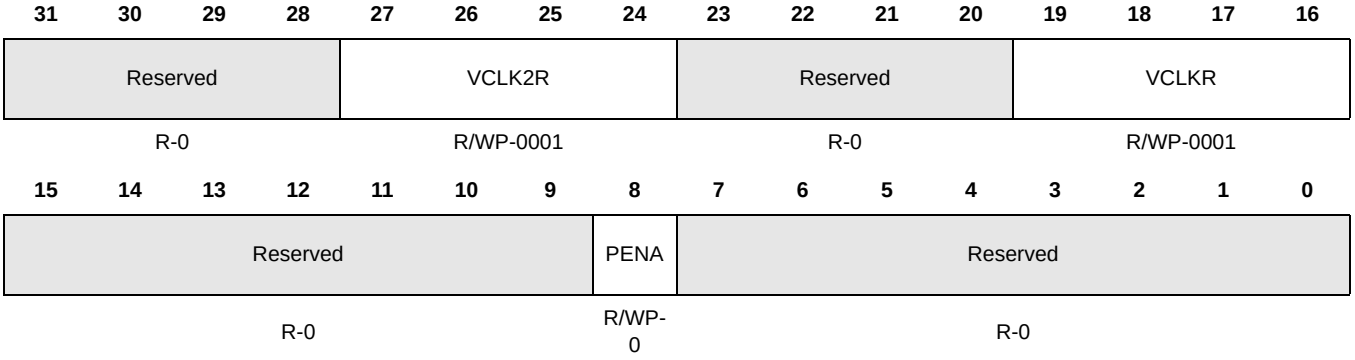
Table 4-48. MMU Global Control Register (MMUGCR) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved		Reads return zero and writes have no effect.
0	MPMENA		This generates CPU reset specific to the CPU only, not to the system (system reset remains inactive during CPU reset). Note: The MPMENA bit is cleared to 0 only on power-up reset.
		0	No CPU reset was generated.
		1	CPU reset was generated since the last PORRST.

4.2.48 Clock Control Register (CLKCNTL)

This register is shown in Figure 4-47 and described in Table 4-49.

Figure 4-47. Clock Control Register (CLKCNTRL) [offset = 0xD0]



R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 4-49. Clock Control Register (CLKCNTRL) Field Descriptions

Bit	Name	Value	Description
31–28	Reserved		Reads return zero and writes have no effect.
27–24	VCLKR2[3:0]		VBUS clock2 ratio.
		0000	The ratio is HCLK divided by 1.
		...	...
		1111	The ratio is HCLK divided by 16.
23–20	Reserved		Reads return zero and writes have no effect.
19–16	VCLKR[3:0]		VBUS clock ratio.
		0000	The ratio is HCLK divide by 1.
		...	...
		1111	The ratio is HCLK divided by 16.
15–9	Reserved		Reads return zero and writes have no effect.
8	PENA		Peripheral enable bit.
		0	The global peripheral/peripheral memory frames are in reset.
		1	All peripheral/peripheral memory frames are out of reset.
7–0	Reserved		Reads return zero and writes have no effect.

4.2.49 ECP Control Register (ECPCNTL)

This register is shown in Figure 4-48 and described in Table 4-50.

Figure 4-48. ECP Control Register (ECPCNTL) [offset = 0xD4]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved							ECP-SSEL	ECP COS	Reserved						
R-0							R/W-0	R/W-0	R-0						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ECPDIV15:0															
R/W-0															

R = Read; WP = Write in privileged mode only; -n = Value after reset

.

Table 4-50. ECP Control Register (ECPCNTL) Field Descriptions

Bit	Name	Value	Description
31–25	Reserved		Reads return zero and writes have no effect.
24	ECPSSEL		ECP Source Clock Select for ECP module.
		0	ECP's source clock is VCLK.
		1	ECP's source clock is oscillator.
23	ECPCOS		ECP continue on suspend.
		0	ECPCLK output is disabled in suspend mode. ECPCLK output will be shut off and will not be seen on the I/O pin of the device.
		1	ECPCLK output is enabled in suspend mode. ECPCLK output will be seen on the I/O pin of the device.
22–16	Reserved		Reads return zero and writes have no effect.
15–0	ECPDIV(15–0)	0–FFFF	ECP divider value. The value of ECPDIV bits determine the external clock (ECP clock) frequency as a ratio of VBUS clock (VCLK) as $ECPCLK = \frac{VCLK}{(ECPDIV + 1)}$ Note: The device does not guarantee ECPCLK beyond 20 MHz because of the slow output buffers used for (EMI) purposes.

4.2.50 DSP Master Global Control Register (DSPGCR)

This register is shown in Figure 4-49 and described in Table 4-51.

Figure 4-49. DSP Master Global Control Register (DSPGCR) [offset = 0xD8]

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

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

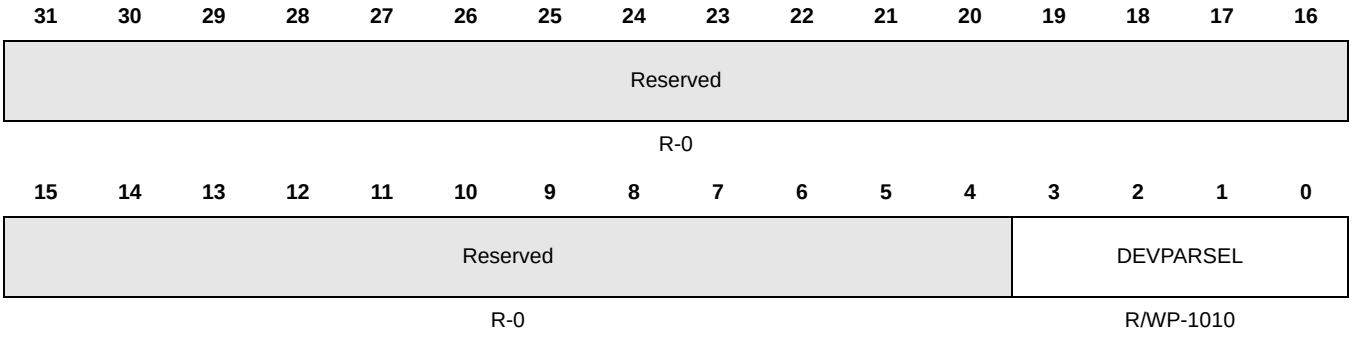
Table 4-51. DSP Master Global Control Register (DSPGCR) Field Descriptions

Bit	Name	Value	Description
31–0	Reserved		Reads return zero and writes have no effect.

4.2.51 DEV Parity Control Register1 (DEVCR1)

This register is shown in Figure 4-50 and described in Table 4-52.

Figure 4-50. DEV Parity Control Register1 (DEVCR1) [offset = 0xDC]



R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 4-52. DEV Parity Control Register1 (DEVCR1) Field Descriptions

Bit	Name	Value	Description
31–4	Reserved		Reads return zero and writes have no effect.
3–0	DEVPARSEL		Device parity select bit key. Note: After an odd (**DEVPARSEL**=0101) or even (**DEVPARSEL**=1010) 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.
		1010	The device parity is odd.
		0101	The device parity is even.

4.2.52 System Exception Control Register (SYSECR)

This register is shown in Figure 4-51 and described in Table 4-51.

Figure 4-51. System Exception Control Register (SYSECR) [offset = 0xE0]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESET1	RESET0	Reserved													
R/WP-0	R/WP-1	R-0													

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 4-53. System Exception Control Register (SYSECR) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Reads return zero and writes have no effect.
15–14	RESET[1:0]		Software reset bits. Setting RESET1 or clearing RESET0 causes a software reset. There are three possible reset cases:
		01	No reset will occur.
		1n, n0	A global system reset will occur.
13–0	Reserved		Reads return zero and writes have no effect.

4.2.53 System Exception Status Register (SYSESR)

This register is shown in Figure 4-52 and described in Table 4-54.

Figure 4-52. System Exception Status Register (SYSESR) [offset = 0xE4]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PO RST	OSC RST	WD RST	Reserved							CPU RST	SW RST	EXT RST	Reserved		
R/WC-X	R/WC-X	R/WC-X	R-0							R/WC-X	R/WC-X	R/WC-X	R-0		

R = Read in all modes; WC = Write clear in all modes; -X = Value unchanged after reset; -n = Value after reset

.

Table 4-54. System Exception Status Register (SYSESR) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Reads return zero and writes have no effect.
15	PORST		Power-up reset. This bit is set when VCCOR (VCC out of range) is asserted. Reset is asserted as long as VCCOR is active.
		0	Read: The V_{CC} has not been out of regulation. Write: The bit is unchanged.
		1	Read: The last reset caused by V_{CC} being out of regulation (nPORST is asserted). Write: The bit is cleared to 0.
14	OSCRST		Reset caused by oscillator failure or PLL cycle slip. This bit is set when a reset is caused by an oscillator failure.
		0	Read: No reset has occurred because of oscillator failure or PLL cycle slip. Write: The bit is unchanged.
		1	Read: The last reset was caused by oscillator failure or PLL cycle slip. Write: The bit is cleared to 0.
13	WDRST		Watchdog and debug reset flag. This bit is set when the last reset was caused by the analog or digital watchdog (AWD or DWD) or the ICEPICK debug reset request.
		0	Read: No reset has occurred. Write: The bit is unchanged.
		1	Read: The last reset was caused by the AWD, DWD, or ICEPICK. Write: The bit is cleared to 0.
12-6	Reserved		Reads return zero and writes have no effect.

Table 4-54. System Exception Status Register (SYSESR) Field Descriptions (Continued)

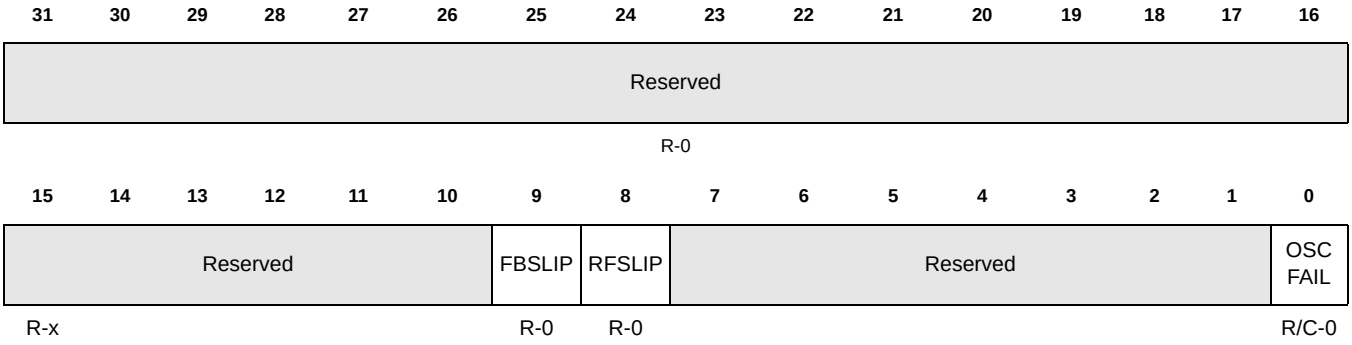
Bit	Name	Value	Description
5	CPURST		CPU reset flag. This bit is set when the last reset to the CPU is caused by: 1) Self test controller reset 2)Switching from BOOT ROM to FLASH by using BTRMSLRST register.
		0	Read: No reset caused by a self test controller reset or BOOT memory change. Write: The bit is unchanged.
		1	Read: The last reset was caused by a BOOT memory change or self test controller reset. Write: The bit is cleared to 0.
4	SWRST		Software reset flag. This bit is set when the last reset is caused by software writing to the RESET1 or RESET0 bits.
		0	Read: No software reset has occurred. Write: The bit is unchanged.
		1	Read: The last reset was a software reset. Write: The bit is cleared to 0.
3	EXTRST		External reset flag. This bit is set when the last reset is caused by the external reset pin or internally generated nRST.
		0	Read: No reset caused by the external reset pin has occurred. Write: The bit is unchanged.
		1	Read: The last reset was caused by the external reset pin or internally generated nRST. Write: The bit is cleared to 0.
2-0	Reserved		Reads return zero and writes have no effect. **NOTE**: If the system is configured to use the MPU, bit 0 of the SYSESR register will be set to a 1.

Note:
EXTRST bit will always be set along with any reset sources with the exception of CPURST. In devices where nRST pin is not present the EXTRST bit is NEVER set.

4.2.54 Global Status Register (GLBSTAT)

This register is shown in Figure 4-53 and described in Table 4-55.

Figure 4-53. Global Status Register (GLBSTAT) [offset = 0xEC]



R = Read; C = Clear;- X = Value changed after reset

.

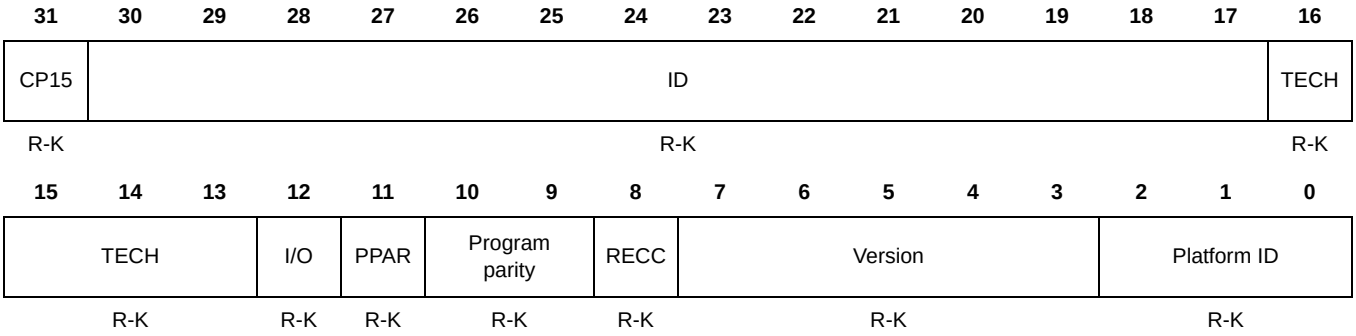
Table 4-55. Global Status Register (GLBSTAT) Field Descriptions

Bit	Name	Value	Description
31–10	Reserved		Reads return zero and writes have no effect.
9	FBSLIP		Over cycle slip detection.
		0	Read: No over cycle slip has been detected. Write: The bit is unchanged.
		1	Read: An over cycle slip has been detected. Write: The bit is cleared to 0.
8	RFSLIP		Under cycle slip detection.
		0	Read: No under cycle slip has been detected. Write: The bit is unchanged.
		1	Read: An under cycle slip has been detected. Write: The bit is cleared to 0.
7-1	Reserved		Reads return zero and writes have no effect.
0	OSCFAIL		Oscillator fail flag bit.
		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.

4.2.55 Device Identification Register (DEVID)

The DEV is a read-only register. It contains device-specific information that is hard-coded during device manufacture.This register is shown in Figure 4-54 and described in Table 4-56.

Figure 4-54. Device Identification Register (DEVID) [offset = 0xF0]



R = Read only; -K = Constant value

Note:
For the constant value of k (see above) refer to the device specific datasheet.

Table 4-56. Device Identification Register (DEVID) Field Descriptions

Bit	Name	Value	Description
31	CP15		CP15 CPU. This bit indicates whether the CPU has a coprocessor 15 (CP15)
		0	The CPU has a CP15 present. The CPU ID can be read using the CP15 C0,C0,0 register.
		1	The CPU has no CP15 present. The automotive platform supports only one CPU without a CP15: the ARM7TDMI. Note: An ARM7TDMI could have a CP15. In this case, the CPU ID will give the information of the CPU.
30–17	ID	0–3FFFh	Device ID. The device ID is unique by device configuration and is defined in the device datasheet.
16–13	TECH		These bits define the process technology by which the device was manufactured.
		0000	The C05 process technology was used.
		0001	The F05 process technology was used.
		0010	The C035 process technology was used.
		0011	The F035 process technology was used.
12	I/O		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.

Table 4-56. Device Identification Register (DEVID) Field Descriptions (Continued)

Bit	Name	Value	Description
11	PPAR		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	Program parity		These bits indicate which parity is present for the program memory.
		00	No memory protection is present.
		01	The program memory has single bit parity.
		10	The program memory has ECC parity.
		11	These bits are reserved.
8	RECC		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–F	Version. These bits provide the revision of the device.
2–0	Platform ID	101	The device is part of the TMS470M Series family.

4.2.56 Software Interrupt Vector Register (SSIVEC)

This register is shown in Figure 4-55 and described in Table 4-57.

Figure 4-55. Software Interrupt Vector Register (SSIVEC) [offset = 0xF4]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SSIDATA								SSIVECT							
R-0								R-0							

R = Read only; -n = Value after reset

.

Table 4-57. Software Interrupt Vector Register (SSIVEC) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Reads return zero and writes have no effect.
15–8	SSIDATA[7:0]	0–FF	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[7:0]		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 scheme: SSIR1 has the highest priority. SSIR4 has the lowest priority.
		0h	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

4.2.57 System Software Interrupt Flag Register (SSIF)

This register is shown in Figure 4-56 and described in Table 4-58.

Figure 4-56. System Software Interrupt Flag Register (SSIF) [offset = 0xF8]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												SSI_FLAG4	SSI_FLAG3	SSI_FLAG2	SS_FLAG1
R-0												R/W-0	R/W-0	R/W-0	R/W-0

R = Read in all modes; W = Write in all modes; -n = Value after reset

.

Table 4-58. System Software Interrupt Flag Register (SSIF) Field Descriptions

Bit	Name	Value	Description
31–4	Reserved		Reads return zero and 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.

4.3 Peripheral Central Resource (PCR) Control Registers

This section describes the PCR control registers. The start address of the PCR register frame is 0xFFFF E000 and the end address is 0xFFFF E0FF. Figure 4-57 summarizes the registers in the peripheral central resource (PCR), which are used to configure protection to the peripherals in PCS and PS regions. The following sections provide detailed information about these registers.

Figure 4-57. PCR Register Summary

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
0x00	PCS 31 PROT SET	PCS 30 PROT SET	PCS 29 PROT SET	PCS 28 PROT SET	PCS 27 PROT SET	PCS 26 PROT SET	PCS 25 PROT SET	PCS 24 PROT SET	PCS 23 PROT SET	PCS 22 PROT SET	PCS 21 PROT SET	PCS 20 PROT SET	PCS 19 PROT SET	PCS 18 PROT SET	PCS 17 PROT SET	PCS 16 PROT SET
PMPROTSET0 Page 148	PCS 15 PROT SET	PCS 14 PROT SET	PCS 13 PROT SET	PCS 12 PROT SET	PCS 11 PROT SET	PCS 10 PROT SET	PCS 9 PROT SET	PCS 8 PROT SET	PCS 7 PROT SET	PCS 6 PROT SET	PCS 5 PROT SET	PCS 4 PROT SET	PCS 3 PROT SET	PCS 2 PROT SET	PCS 1 PROT SET	PCS 0 PROT SET
0x04	PCS 63 PROT SET	PCS 62 PROT SET	PCS 61 PROT SET	PCS 60 PROT SET	PCS 59 PROT SET	PCS 58 PROT SET	PCS 57 PROT SET	PCS 56 PROT SET	PCS 55 PROT SET	PCS 54 PROT SET	PCS 53 PROT SET	PCS 52 PROT SET	PCS 51 PROT SET	PCS 50 PROT SET	PCS 49 PROT SET	PCS 48 PROT SET
PMPROTSET1 Page 149	PCS 47 PROT SET	PCS 46 PROT SET	PCS 45 PROT SET	PCS 44 PROT SET	PCS 43 PROT SET	PCS 42 PROT SET	PCS 41 PROT SET	PCS 40 PROT SET	PCS 39 PROT SET	PCS 38 PROT SET	PCS 37 PROT SET	PCS 36 PROT SET	PCS 35 PROT SET	PCS 34 PROT SET	PCS 33 PROT SET	PCS 32 PROT SET
0x08	Reserved															
Reserved	Reserved															
0x0C	Reserved															
Reserved	Reserved															
0x10	PCS 31 PROT CLR	PCS 30 PROT CLR	PCS 29 PROT CLR	PCS 28 PROT CLR	PCS 27 PROT CLR	PCS 26 PROT CLR	PCS 25 PROT CLR	PCS 24 PROT CLR	PCS 23 PROT CLR	PCS 22 PROT CLR	PCS 21 PROT CLR	PCS 20 PROT CLR	PCS 19 PROT CLR	PCS 18 PROT CLR	PCS 17 PROT CLR	PCS 16 PROT CLR
PMPROTCLR0 Page 150	PCS 15 PROT CLR	PCS 14 PROT CLR	PCS 13 PROT CLR	PCS 12 PROT CLR	PCS 11 PROT CLR	PCS 10 PROT CLR	PCS 9 PROT CLR	PCS 8 PROT CLR	PCS 7 PROT CLR	PCS 6 PROT CLR	PCS 5 PROT CLR	PCS 4 PROT CLR	PCS 3 PROT CLR	PCS 2 PROT CLR	PCS 1 PROT CLR	PCS 0 PROT CLR
0x14	PCS 63 PROT CLR	PCS 62 PROT CLR	PCS61 PROT CLR	PCS 60 PROT CLR	PCS 59 PROT CLR	PCS 58 PROT CLR	PCS 57 PROT CLR	PCS 56 PROT CLR	PCS 55 PROT CLR	PCS 54 PROT CLR	PCS 53 PROT CLR	PCS 52 PROT CLR	PCS 51 PROT CLR	PCS 50 PROT CLR	PCS 49 PROT CLR	PCS 48 PROT CLR
PMPROTCLR1 Page 151	PCS47 PROT CLR	PCS46 PROT CLR	PCS45 PROT CLR	PCS44 PROT CLR	PCS43 PROT CLR	PCS42 PROT CLR	PCS41 PROT CLR	PCS40 PROT CLR	PCS39 PROT CLR	PCS38 PROT CLR	PCS37 PROT CLR	PCS36 PROT CLR	PCS35 PROT CLR	PCS34 PROT CLR	PCS33 PROT CLR	PCS32 PROT CLR

Figure 4-57. PCR Register Summary (Continued)

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
0x18	Reserved															
Reserved	Reserved															
0x1C	Reserved															
Reserved	Reserved															
0x20	PS7 QUAD 3 PROT SET	PS7 QUAD 2 PROT SET	PS7 QUAD 1 PROT SET	PS7 QUAD 0 PROT SET	PS6 QUAD 3 PROT SET	PS6 QUAD 2 PROT SET	PS6 QUAD 1 PROT SET	PS6 QUAD 0 PROT SET	PS5 QUAD 3 PROT SET	PS5 QUAD 2 PROT SET	PS5 QUAD 1 PROT SET	PS5 QUAD 0 PROT SET	PS4 QUAD 3 PROT SET	PS4 QUAD 2 PROT SET	PS4 QUAD 1 PROT SET	PS4 QUAD 0 PROT SET
PPROTSET0 Page 152	PS3 QUAD 3 PROT SET	PS3 QUAD 2 PROT SET	PS3 QUAD 1 PROT SET	PS3 QUAD 0 PROT SET	PS2 QUAD 3 PROT SET	PS2 QUAD 2 PROT SET	PS2 QUAD 1 PROT SET	PS2 QUAD 0 PROT SET	PS1 QUAD 3 PROT SET	PS1 QUAD 2 PROT SET	PS1 QUAD 1 PROT SET	PS1 QUAD 0 PROT SET	PS0 QUAD 3 PROT SET	PS0 QUAD 2 PROT SET	PS0 QUAD 1 PROT SET	PS0 QUAD 0 PROT SET
0x24	PS15 QUAD 3 PROT SET	PS15 QUAD 2 PROT SET	PS15 QUAD 1 PROT SET	PS15 QUAD 0 PROT SET	PS14 QUAD 3 PROT SET	PS14 QUAD 2 PROT SET	PS14 QUAD 1 PROT SET	PS14 QUAD 0 PROT SET	PS13 QUAD 3 PROT SET	PS13 QUAD 2 PROT SET	PS13 QUAD 1 PROT SET	PS13 QUAD 0 PROT SET	PS12 QUAD 3 PROT SET	PS12 QUAD 2 PROT SET	PS12 QUAD 1 PROT SET	PS12 QUAD 0 PROT SET
PPROTSET1 Page 154	PS11 QUAD 3 PROT SET	PS11 QUAD 2 PROT SET	PS11 QUAD 1 PROT SET	PS11 QUAD 0 PROT SET	PS10 QUAD 3 PROT SET	PS10 QUAD 2 PROT SET	PS10 QUAD 1 PROT SET	PS10 QUAD 0 PROT SET	PS9 QUAD 3 PROT SET	PS9 QUAD 2 PROT SET	PS9 QUAD 1 PROT SET	PS9 QUAD 0 PROT SET	PS8 QUAD 3 PROT SET	PS8 QUAD 2 PROT SET	PS8 QUAD 1 PROT SET	PS8 QUAD 0 PROT SET
0x28	PS23 QUAD 3 PROT SET	PS23 QUAD 2 PROT SET	PS23 QUAD 1 PROT SET	PS23 QUAD 0 PROT SET	PS22 QUAD 3 PROT SET	PS22 QUAD 2 PROT SET	PS22 QUAD 1 PROT SET	PS22 QUAD 0 PROT SET	PS21 QUAD 3 PROT SET	PS21 QUAD 2 PROT SET	PS21 QUAD 1 PROT SET	PS21 QUAD 0 PROT SET	PS20 QUAD 3 PROT SET	PS20 QUAD 2 PROT SET	PS20 QUAD 1 PROT SET	PS20 QUAD 0 PROT SET
PPROTSET2 Page 155	PS19 QUAD 3 PROT SET	PS19 QUAD 2 PROT SET	PS19 QUAD 1 PROT SET	PS19 QUAD 0 PROT SET	PS18 QUAD 3 PROT SET	PS18 QUAD 2 PROT SET	PS18 QUAD 1 PROT SET	PS18 QUAD 0 PROT SET	PS17 QUAD 3 PROT SET	PS17 QUAD 2 PROT SET	PS17 QUAD 1 PROT SET	PS17 QUAD 0 PROT SET	PS16 QUAD 3 PROT SET	PS16 QUAD 2 PROT SET	PS16 QUAD 1 PROT SET	PS16 QUAD 0 PROT SET
0x2C	PS31 QUAD 3 PROT SET	PS31 QUAD 2 PROT SET	PS31 QUAD 1 PROT SET	PS31 QUAD 0 PROT SET	PS30 QUAD 3 PROT SET	PS30 QUAD 2 PROT SET	PS30 QUAD 1 PROT SET	PS30 QUAD 0 PROT SET	PS29 QUAD 3 PROT SET	PS29 QUAD 2 PROT SET	PS29 QUAD 1 PROT SET	PS29 QUAD 0 PROT SET	PS28 QUAD 3 PROT SET	PS28 QUAD 2 PROT SET	PS28 QUAD 1 PROT SET	PS28 QUAD 0 PROT SET
PPROTSET3 Page 156	PS27 QUAD 3 PROT SET	PS27 QUAD 2 PROT SET	PS27 QUAD 1 PROT SET	PS27 QUAD 0 PROT SET	PS26 QUAD 3 PROT SET	PS26 QUAD 2 PROT SET	PS26 QUAD 1 PROT SET	PS26 QUAD 0 PROT SET	PS25 QUAD 3 PROT SET	PS25 QUAD 2 PROT SET	PS25 QUAD 1 PROT SET	PS25 QUAD 0 PROT SET	PS24 QUAD 3 PROT SET	PS24 QUAD 2 PROT SET	PS24 QUAD 1 PROT SET	PS24 QUAD 0 PROT SET

Figure 4-57. PCR Register Summary (Continued)

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
0x30	Reserved															
Reserved	Reserved															
0x34	Reserved															
Reserved	Reserved															
0x38	Reserved															
Reserved	Reserved															
0x3C	Reserved															
Reserved	Reserved															
0x40	PS7 QUAD 3 PROT CLR	PS7 QUAD 2 PROT CLR	PS7 QUAD 1 PROT CLR	PS7 QUAD 0 PROT CLR	PS6 QUAD 3 PROT CLR	PS6 QUAD 2 PROT CLR	PS6 QUAD 1 PROT CLR	PS6 QUAD 0 PROT CLR	PS5 QUAD 3 PROT CLR	PS5 QUAD 2 PROT CLR	PS5 QUAD 1 PROT CLR	PS5 QUAD 0 PROT CLR	PS4 QUAD 3 PROT CLR	PS4 QUAD 2 PROT CLR	PS4 QUAD 1 PROT CLR	PS4 QUAD 0 PROT CLR
PPROTCLR0 Page 157	PS3 QUAD 3 PROT CLR	PS3 QUAD 2 PROT CLR	PS3 QUAD 1 PROT CLR	PS3 QUAD 0 PROT CLR	PS2 QUAD 3 PROT CLR	PS2 QUAD 2 PROT CLR	PS2 QUAD 1 PROT CLR	PS2 QUAD 0 PROT CLR	PS1 QUAD 3 PROT CLR	PS1 QUAD 2 PROT CLR	PS1 QUAD 1 PROT CLR	PS1 QUAD 0 PROT CLR	PS0 QUAD 3 PROT CLR	PS0 QUAD 2 PROT CLR	PS0 QUAD 1 PROT CLR	PS0 QUAD 0 PROT CLR
0x44	PS15 QUAD 3 PROT CLR	PS15 QUAD 2 PROT CLR	PS15 QUAD 1 PROT CLR	PS15 QUAD 0 PROT CLR	PS14 QUAD 3 PROT CLR	PS14 QUAD 2 PROT CLR	PS14 QUAD 1 PROT CLR	PS14 QUAD 0 PROT CLR	PS13 QUAD 3 PROT CLR	PS13 QUAD 2 PROT CLR	PS13 QUAD 1 PROT CLR	PS13 QUAD 0 PROT CLR	PS12 QUAD 3 PROT CLR	PS12 QUAD 2 PROT CLR	PS12 QUAD 1 PROT CLR	PS12 QUAD 0 PROT CLR
PPROTCLR1 Page 158	PS11 QUAD 3 PROT CLR	PS11 QUAD 2 PROT CLR	PS11 QUAD 1 PROT CLR	PS11 QUAD 0 PROT CLR	PS10 QUAD 3 PROT CLR	PS10 QUAD 2 PROT CLR	PS10 QUAD 1 PROT CLR	PS10 QUAD 0 PROT CLR	PS9 QUAD 3 PROT CLR	PS9 QUAD 2 PROT CLR	PS9 QUAD 1 PROT CLR	PS9 QUAD 0 PROT CLR	PS8 QUAD 3 PROT CLR	PS8 QUAD 2 PROT CLR	PS8 QUAD 1 PROT CLR	PS8 QUAD 0 PROT CLR

Figure 4-57. PCR Register Summary (Continued)

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
0x48	PS23 QUAD 3 PROT CLR	PS23 QUAD 2 PROT CLR	PS23 QUAD 1 PROT CLR	PS23 QUAD 0 PROT CLR	PS22 QUAD 3 PROT CLR	PS22 QUAD 2 PROT CLR	PS22 QUAD 1 PROT CLR	PS22 QUAD 0 PROT CLR	PS21 QUAD 3 PROT CLR	PS21 QUAD 2 PROT CLR	PS21 QUAD 1 PROT CLR	PS21 QUAD 0 PROT CLR	PS20 QUAD 3 PROT CLR	PS20 QUAD 2 PROT CLR	PS20 QUAD 1 PROT CLR	PS20 QUAD 0 PROT CLR
PPROTCLR2 Page 159	PS19 QUAD 3 PROT CLR	PS19 QUAD 2 PROT CLR	PS19 QUAD 1 PROT CLR	PS19 QUAD 0 PROT CLR	PS18 QUAD 3 PROT CLR	PS18 QUAD 2 PROT CLR	PS18 QUAD 1 PROT CLR	PS18 QUAD 0 PROT CLR	PS17 QUAD 3 PROT CLR	PS17 QUAD 2 PROT CLR	PS17 QUAD 1 PROT CLR	PS17 QUAD 0 PROT CLR	PS16 QUAD 3 PROT CLR	PS16 QUAD 2 PROT CLR	PS16 QUAD 1 PROT CLR	PS16 QUAD 0 PROT CLR
0x4C	PS31 QUAD 3 PROT CLR	PS31 QUAD 2 PROT CLR	PS31 QUAD 1 PROT CLR	PS31 QUAD 0 PROT CLR	PS30 QUAD 3 PROT CLR	PS30 QUAD 2 PROT CLR	PS30 QUAD 1 PROT CLR	PS30 QUAD 0 PROT CLR	PS29 QUAD 3 PROT CLR	PS29 QUAD 2 PROT CLR	PS29 QUAD 1 PROT CLR	PS29 QUAD 0 PROT CLR	PS28 QUAD 3 PROT CLR	PS28 QUAD 2 PROT CLR	PS28 QUAD 1 PROT CLR	PS28 QUAD 0 PROT CLR
PPROTCLR3 Page 160	PS27 QUAD 3 PROT CLR	PS27 QUAD 2 PROT CLR	PS27 QUAD 1 PROT CLR	PS27 QUAD 0 PROT CLR	PS26 QUAD 3 PROT CLR	PS26 QUAD 2 PROT CLR	PS26 QUAD 1 PROT CLR	PS26 QUAD 0 PROT CLR	PS25 QUAD 3 PROT CLR	PS25 QUAD 2 PROT CLR	PS25 QUAD 1 PROT CLR	PS25 QUAD 0 PROT CLR	PS24 QUAD 3 PROT CLR	PS24 QUAD 2 PROT CLR	PS24 QUAD 1 PROT CLR	PS24 QUAD 0 PROT CLR
0x50	Reserved															
Reserved	Reserved															
0x54	Reserved															
Reserved	Reserved															
0x58	Reserved															
Reserved	Reserved															
0x5C	Reserved															
Reserved	Reserved															
0x60	PCS31 PWR DWN SET	PCS30 PWR DWN SET	PCS29 PWR DWN SET	PCS28 PWR DWN SET	PCS27 PWR DWN SET	PCS26 PWR DWN SET	PCS25 PWR DWN SET	PCS24 PWR DWN SET	PCS23 PWR DWN SET	PCS22 PWR DWN SET	PCS21 PWR DWN SET	PCS20 PWR DWN SET	PCS19 PWR DWN SET	PCS18 PWR DWN SET	PCS17 PWR DWN SET	PCS16 PWR DWN SET
PCSPWRDWN SET0 Page 161	PCS15 PWR DWN SET	PCS14 PWR DWN SET	PCS13 PWR DWN SET	PCS12 PWR DWN SET	PCS11 PWR DWN SET	PCS10 PWR DWN SET	PCS9 PWR DWN SET	PCS8 PWR DWN SET	PCS7 PWR DWN SET	PCS6 PWR DWN SET	PCS5 PWR DWN SET	PCS4 PWR DWN SET	PCS3 PWR DWN SET	PCS2 PWR DWN SET	PCS1 PWR DWN SET	PCS0 PWR DWN SET

Figure 4-57. PCR Register Summary (Continued)

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
0x64 PCSPWRDWNSET1 Page 162	PCS63 PWRDWNSET	PCS62 PWRDWNSET	PCS61 PWRDWNSET	PCS60 PWRDWNSET	PCS59 PWRDWNSET	PCS58 PWRDWNSET	PCS57 PWRDWNSET	PCS56 PWRDWNSET	PCS55 PWRDWNSET	PCS54 PWRDWNSET	PCS53 PWRDWNSET	PCS52 PWRDWNSET	PCS51 PWRDWNSET	PCS50 PWRDWNSET	PCS49 PWRDWNSET	PCS48 PWRDWNSET
	PCS47 PWRDWNSET	PCS46 PWRDWNSET	PCS45 PWRDWNSET	PCS44 PWRDWNSET	PCS43 PWRDWNSET	PCS42 PWRDWNSET	PCS41 PWRDWNSET	PCS40 PWRDWNSET	PCS39 PWRDWNSET	PCS38 PWRDWNSET	PCS37 PWRDWNSET	PCS36 PWRDWNSET	PCS35 PWRDWNSET	PCS34 PWRDWNSET	PCS33 PWRDWNSET	PCS32 PWRDWNSET
0x68 Reserved	Reserved															
	Reserved															
0x6C Reserved	Reserved															
	Reserved															
0x70 PCSPWRDWNCLR0 Page 163	PCS31 PWRDWNCLR	PCS30 PWRDWNCLR	PCS29 PWRDWNCLR	PCS28 PWRDWNCLR	PCS27 PWRDWNCLR	PCS26 PWRDWNCLR	PCS25 PWRDWNCLR	PCS24 PWRDWNCLR	PCS23 PWRDWNCLR	PCS22 PWRDWNCLR	PCS21 PWRDWNCLR	PCS20 PWRDWNCLR	PCS19 PWRDWNCLR	PCS18 PWRDWNCLR	PCS17 PWRDWNCLR	PCS16 PWRDWNCLR
	PCS15 PWRDWNCLR	PCS14 PWRDWNCLR	PCS13 PWRDWNCLR	PCS12 PWRDWNCLR	PCS11 PWRDWNCLR	PCS10 PWRDWNCLR	PCS9 PWRDWNCLR	PCS8 PWRDWNCLR	PCS7 PWRDWNCLR	PCS6 PWRDWNCLR	PCS5 PWRDWNCLR	PCS4 PWRDWNCLR	PCS3 PWRDWNCLR	PCS2 PWRDWNCLR	PCS1 PWRDWNCLR	PCS0 PWRDWNCLR
0x74 PCSPWRDWNCLR1 Page 164	PCS63 PWRDWNCLR	PCS62 PWRDWNCLR	PCS61 PWRDWNCLR	PCS60 PWRDWNCLR	PCS59 PWRDWNCLR	PCS58 PWRDWNCLR	PCS57 PWRDWNCLR	PCS56 PWRDWNCLR	PCS55 PWRDWNCLR	PCS54 PWRDWNCLR	PCS53 PWRDWNCLR	PCS52 PWRDWNCLR	PCS51 PWRDWNCLR	PCS50 PWRDWNCLR	PCS49 PWRDWNCLR	PCS48 PWRDWNCLR
	PCS47 PWRDWNCLR	PCS46 PWRDWNCLR	PCS45 PWRDWNCLR	PCS44 PWRDWNCLR	PCS43 PWRDWNCLR	PCS42 PWRDWNCLR	PCS41 PWRDWNCLR	PCS40 PWRDWNCLR	PCS39 PWRDWNCLR	PCS38 PWRDWNCLR	PCS37 PWRDWNCLR	PCS36 PWRDWNCLR	PCS35 PWRDWNCLR	PCS34 PWRDWNCLR	PCS33 PWRDWNCLR	PCS32 PWRDWNCLR
0x78 Reserved	Reserved															
	Reserved															

Figure 4-57. PCR Register Summary (Continued)

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
0x7C Reserved	Reserved															
	Reserved															
0x80 PSPWRDWNSET0 Page 165	PS7 QUAD 3 PWR DWN SET	PS7 QUAD 2 PWR DWN SET	PS7 QUAD 1 PWR DWN SET	PS7 QUAD 0 PWR DWN SET	PS6 QUAD 3 PWR DWN SET	PS6 QUAD 2 PWR DWN SET	PS6 QUAD 1 PWR DWN SET	PS6 QUAD 0 PWR DWN SET	PS5 QUAD 3 PWR DWN SET	PS5 QUAD 2 PWR DWN SET	PS5 QUAD 1 PWR DWN SET	PS5 QUAD 0 PWR DWN SET	PS4 QUAD 3 PWR DWN SET	PS4 QUAD 2 PWR DWN SET	PS4 QUAD 1 PWR DWN SET	PS4 QUAD 0 PWR DWN SET
	PS3 QUAD 3 PWR DWN SET	PS3 QUAD 2 PWR DWN SET	PS3 QUAD 1 PWR DWN SET	PS3 QUAD 0 PWR DWN SET	PS2 QUAD 3 PWR DWN SET	PS2 QUAD 2 PWR DWN SET	PS2 QUAD 1 PWR DWN SET	PS2 QUAD 0 PWR DWN SET	PS1 QUAD 3 PWR DWN SET	PS1 QUAD 2 PWR DWN SET	PS1 QUAD 1 PWR DWN SET	PS1 QUAD 0 PWR DWN SET	PS0 QUAD 3 PWR DWN SET	PS0 QUAD 2 PWR DWN SET	PS0 QUAD 1 PWR DWN SET	PS0 QUAD 0 PWR DWN SET
0x84 PSPWRDWNSET1 Page 167	PS15 QUAD 3 PWR DWN SET	PS15 QUAD 2 PWR DWN SET	PS15 QUAD 1 PWR DWN SET	PS15 QUAD 0 PWR DWN SET	PS14 QUAD 3 PWR DWN SET	PS14 QUAD 2 PWR DWN SET	PS14 QUAD 1 PWR DWN SET	PS14 QUAD 0 PWR DWN SET	PS13 QUAD 3 PWR DWN SET	PS13 QUAD 2 PWR DWN SET	PS13 QUAD 1 PWR DWN SET	PS13 QUAD 0 PWR DWN SET	PS12 QUAD 3 PWR DWN SET	PS12 QUAD 2 PWR DWN SET	PS12 QUAD 1 PWR DWN SET	PS12 QUAD 0 PWR DWN SET
	PS11 QUAD 3 PWR DWN SET	PS11 QUAD 2 PWR DWN SET	PS11 QUAD 1 PWR DWN SET	PS11 QUAD 0 PWR DWN SET	PS10 QUAD 3 PWR DWN SET	PS10 QUAD 2 PWR DWN SET	PS10 QUAD 1 PWR DWN SET	PS10 QUAD 0 PWR DWN SET	PS9 QUAD 3 PWR DWN SET	PS9 QUAD 2 PWR DWN SET	PS9 QUAD 1 PWR DWN SET	PS9 QUAD 0 PWR DWN SET	PS8 QUAD 3 PWR DWN SET	PS8 QUAD 2 PWR DWN SET	PS8 QUAD 1 PWR DWN SET	PS8 QUAD 0 PWR DWN SET
0x88 PSPWRDWNSET2 Page 167	PS23 QUAD 3 PWR DWN SET	PS23 QUAD 2 PWR DWN SET	PS23 QUAD 1 PWR DWN SET	PS23 QUAD 0 PWR DWN SET	PS22 QUAD 3 PWR DWN SET	PS22 QUAD 2 PWR DWN SET	PS22 QUAD 1 PWR DWN SET	PS22 QUAD 0 PWR DWN SET	PS21 QUAD 3 PWR DWN SET	PS21 QUAD 2 PWR DWN SET	PS21 QUAD 1 PWR DWN SET	PS21 QUAD 0 PWR DWN SET	PS20 QUAD 3 PWR DWN SET	PS20 QUAD 2 PWR DWN SET	PS20 QUAD 1 PWR DWN SET	PS20 QUAD 0 PWR DWN SET
	PS19 QUAD 3 PWR DWN SET	PS19 QUAD 2 PWR DWN SET	PS19 QUAD 1 PWR DWN SET	PS19 QUAD 0 PWR DWN SET	PS18 QUAD 3 PWR DWN SET	PS18 QUAD 2 PWR DWN SET	PS18 QUAD 1 PWR DWN SET	PS18 QUAD 0 PWR DWN SET	PS17 QUAD 3 PWR DWN SET	PS17 QUAD 2 PWR DWN SET	PS17 QUAD 1 PWR DWN SET	PS17 QUAD 0 PWR DWN SET	PS16 QUAD 3 PWR DWN SET	PS16 QUAD 2 PWR DWN SET	PS16 QUAD 1 PWR DWN SET	PS16 QUAD 0 PWR DWN SET
0x8C PSPWRDWNSET3 Page 169	PS31 QUAD 3 PWR DWN SET	PS31 QUAD 2 PWR DWN SET	PS31 QUAD 1 PWR DWN SET	PS31 QUAD 0 PWR DWN SET	PS30 QUAD 3 PWR DWN SET	PS30 QUAD 2 PWR DWN SET	PS30 QUAD 1 PWR DWN SET	PS30 QUAD 0 PWR DWN SET	PS29 QUAD 3 PWR DWN SET	PS29 QUAD 2 PWR DWN SET	PS29 QUAD 1 PWR DWN SET	PS29 QUAD 0 PWR DWN SET	PS28 QUAD 3 PWR DWN SET	PS28 QUAD 2 PWR DWN SET	PS28 QUAD 1 PWR DWN SET	PS28 QUAD 0 PWR DWN SET
	PS27 QUAD 3 PWR DWN SET	PS27 QUAD 2 PWR DWN SET	PS27 QUAD 1 PWR DWN SET	PS27 QUAD 0 PWR DWN SET	PS26 QUAD 3 PWR DWN SET	PS26 QUAD 2 PWR DWN SET	PS26 QUAD 1 PWR DWN SET	PS26 QUAD 0 PWR DWN SET	PS25 QUAD 3 PWR DWN SET	PS25 QUAD 2 PWR DWN SET	PS25 QUAD 1 PWR DWN SET	PS25 QUAD 0 PWR DWN SET	PS24 QUAD 3 PWR DWN SET	PS24 QUAD 2 PWR DWN SET	PS24 QUAD 1 PWR DWN SET	PS24 QUAD 0 PWR DWN SET

Figure 4-57. PCR Register Summary (Continued)

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 Reserved	Reserved															
	Reserved															
0x94 Reserved	Reserved															
	Reserved															
0x98 Reserved	Reserved															
	Reserved															
0x9C Reserved	Reserved															
	Reserved															
0xA0 PSPWRDWNCLR0 Page 170	PS7 QUAD 3 PWR DWN CLR	PS7 QUAD 2 PWR DWN CLR	PS7 QUAD 1 PWR DWN CLR	PS7 QUAD 0 PWR DWN CLR	PS6 QUAD 3 PWR DWN CLR	PS6 QUAD 2 PWR DWN CLR	PS6 QUAD 1 PWR DWN CLR	PS6 QUAD 0 PWR DWN CLR	PS5 QUAD 3 PWR DWN CLR	PS5 QUAD 2 PWR DWN CLR	PS5 QUAD 1 PWR DWN CLR	PS5 QUAD 0 PWR DWN CLR	PS4 QUAD 3 PWR DWN CLR	PS4 QUAD 2 PWR DWN CLR	PS4 QUAD 1 PWR DWN CLR	PS4 QUAD 0 PWR DWN CLR
	PS3 QUAD 3 PWR DWN CLR	PS3 QUAD 2 PWR DWN CLR	PS3 QUAD 1 PWR DWN CLR	PS3 QUAD 0 PWR DWN CLR	PS2 QUAD 3 PWR DWN CLR	PS2 QUAD 2 PWR DWN CLR	PS2 QUAD 1 PWR DWN CLR	PS2 QUAD 0 PWR DWN CLR	PS1 QUAD 3 PWR DWN CLR	PS1 QUAD 2 PWR DWN CLR	PS1 QUAD 1 PWR DWN CLR	PS1 QUAD 0 PWR DWN CLR	PS0 QUAD 3 PWR DWN CLR	PS0 QUAD 2 PWR DWN CLR	PS0 QUAD 1 PWR DWN CLR	PS0 QUAD 0 PWR DWN CLR
0xA4 PSPWRDWNCLR1 Page 171	PS15 QUAD 3 PWR DWN CLR	PS15 QUAD 2 PWR DWN CLR	PS15 QUAD 1 PWR DWN CLR	PS15 QUAD 0 PWR DWN CLR	PS14 QUAD 3 PWR DWN CLR	PS14 QUAD 2 PWR DWN CLR	PS14 QUAD 1 PWR DWN CLR	PS14 QUAD 0 PWR DWN CLR	PS13 QUAD 3 PWR DWN CLR	PS13 QUAD 2 PWR DWN CLR	PS13 QUAD 1 PWR DWN CLR	PS13 QUAD 0 PWR DWN CLR	PS12 QUAD 3 PWR DWN CLR	PS12 QUAD 2 PWR DWN CLR	PS12 QUAD 1 PWR DWN CLR	PS12 QUAD 0 PWR DWN CLR
	PS11 QUAD 3 PWR DWN CLR	PS11 QUAD 2 PWR DWN CLR	PS11 QUAD 1 PWR DWN CLR	PS11 QUAD 0 PWR DWN CLR	PS10 QUAD 3 PWR DWN CLR	PS10 QUAD 2 PWR DWN CLR	PS10 QUAD 1 PWR DWN CLR	PS10 QUAD 0 PWR DWN CLR	PS9 QUAD 3 PWR DWN CLR	PS9 QUAD 2 PWR DWN CLR	PS9 QUAD 1 PWR DWN CLR	PS9 QUAD 0 PWR DWN CLR	PS8 QUAD 3 PWR DWN CLR	PS8 QUAD 2 PWR DWN CLR	PS8 QUAD 1 PWR DWN CLR	PS8 QUAD 0 PWR DWN CLR

Figure 4-57. PCR Register Summary (Continued)

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
0xA8 PSPWRDWNCLR2 Page 172	PS23QUAD3PWRDWNCLR	PS23QUAD2PWRDWNCLR	PS23QUAD1PWRDWNCLR	PS23QUAD0PWRDWNCLR	PS22QUAD3PWRDWNCLR	PS22QUAD2PWRDWNCLR	PS22QUAD1PWRDWNCLR	PS22QUAD0PWRDWNCLR	PS21QUAD3PWRDWNCLR	PS21QUAD2PWRDWNCLR	PS21QUAD1PWRDWNCLR	PS21QUAD0PWRDWNCLR	PS20QUAD3PWRDWNCLR	PS20QUAD2PWRDWNCLR	PS20QUAD1PWRDWNCLR	PS20QUAD0PWRDWNCLR
	PS19QUAD3PWRDWNCLR	PS19QUAD2PWRDWNCLR	PS19QUAD1PWRDWNCLR	PS19QUAD0PWRDWNCLR	PS18QUAD3PWRDWNCLR	PS18QUAD2PWRDWNCLR	PS18QUAD1PWRDWNCLR	PS18QUAD0PWRDWNCLR	PS17QUAD3PWRDWNCLR	PS17QUAD2PWRDWNCLR	PS17QUAD1PWRDWNCLR	PS17QUAD0PWRDWNCLR	PS16QUAD3PWRDWNCLR	PS16QUAD2PWRDWNCLR	PS16QUAD1PWRDWNCLR	PS16QUAD0PWRDWNCLR
0xAC PSPWRDWNCLR3 Page 173	PS31QUAD3PWRDWNCLR	PS31QUAD2PWRDWNCLR	PS31QUAD1PWRDWNCLR	PS31QUAD0PWRDWNCLR	PS30QUAD3PWRDWNCLR	PS30QUAD2PWRDWNCLR	PS30QUAD1PWRDWNCLR	PS30QUAD0PWRDWNCLR	PS29QUAD3PWRDWNCLR	PS29QUAD2PWRDWNCLR	PS29QUAD1PWRDWNCLR	PS29QUAD0PWRDWNCLR	PS28QUAD3PWRDWNCLR	PS28QUAD2PWRDWNCLR	PS28QUAD1PWRDWNCLR	PS28QUAD0PWRDWNCLR
	PS27QUAD3PWRDWNCLR	PS27QUAD2PWRDWNCLR	PS27QUAD1PWRDWNCLR	PS27QUAD0PWRDWNCLR	PS26QUAD3PWRDWNCLR	PS26QUAD2PWRDWNCLR	PS26QUAD1PWRDWNCLR	PS26QUAD0PWRDWNCLR	PS25QUAD3PWRDWNCLR	PS25QUAD2PWRDWNCLR	PS25QUAD1PWRDWNCLR	PS25QUAD0PWRDWNCLR	PS24QUAD3PWRDWNCLR	PS24QUAD2PWRDWNCLR	PS24QUAD1PWRDWNCLR	PS24QUAD0PWRDWNCLR

4.3.1 Peripheral Memory Protection Set Register 0 (PMPROTSET0)

This register is shown in Figure 4-58 and described in Table 4-59.

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

Figure 4-58. Peripheral Memory Protection Set Register 0 (PMPROTSET0) [offset = 0x00]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PCS31 PROT SET	PCS30 PROT SET	PCS29 PROT SET	PCS28 PROT SET	PCS27 PROT SET	PCS26 PROT SET	PCS25 PROT SET	PCS24 PROT SET	PCS23 PROT SET	PCS22 PROT SET	PCS21 PROT SET	PCS20 PROT SET	PCS19 PROT SET	PCS18 PROT SET	PCS17 PROT SET	PCS16 PROT SET
R/WP-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCS15 PROT SET	PCS14 PROT SET	PCS13 PROT SET	PCS12 PROT SET	PCS11 PROT SET	PCS10 PROT SET	PCS9 PROT SET	PCS8 PROT SET	PCS7 PROT SET	PCS6 PROT SET	PCS5 PROT SET	PCS4 PROT SET	PCS3 PROT SET	PCS2 PROT SET	PCS1 PROT SET	PCS0 PROT SET
R/WP-0															

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 4-59. Peripheral Memory Protection Set Register 0 (PMPROTSET0) Field Descriptions

Bit	Name	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.

4.3.2 Peripheral Memory Protection Set Register1 (PMPROTSET1)

This register is shown in Figure 4-59 and described in Table 4-60.

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

Figure 4-59. Peripheral Memory Protection Set Register1 (PMPROTSET1) [offset = 0x04]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PCS63 PROT SET	PCS62 PROT SET	PCS61 PROT SET	PCS60 PROT SET	PCS59 PROT SET	PCS58 PROT SET	PCS57 PROT SET	PCS56 PROT SET	PCS55 PROT SET	PCS54 PROT SET	PCS53 PROT SET	PCS52 PROT SET	PCS51 PROT SET	PCS50 PROT SET	PCS49 PROT SET	PCS48 PROT SET
R/WP-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCS47 PROT SET	PCS46 PROT SET	PCS45 PROT SET	PCS44 PROT SET	PCS43 PROT SET	PCS42 PROT SET	PCS41 PROT SET	PCS40 PROT SET	PCS39 PROT SET	PCS38 PROT SET	PCS37 PROT SET	PCS36 PROT SET	PCS35 PROT SET	PCS34 PROT SET	PCS33 PROT SET	PCS32 PROT SET
R/WP-0															

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 4-60. Peripheral Memory Protection Set Register1 (PMPROTSET1) Field Descriptions

Bit	Name	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 PMPROTSET0 and PMPROTCLR0 registers is set to 1.

4.3.3 Peripheral Memory Protection Clear Register 0 (PMPROTCLR0)

This register is shown in Figure 4-60 and described in Table 4-61.

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

Figure 4-60. Peripheral Memory Protection Clear Register 0 (PMPROTCLR0) [offset = 0x10]

30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PCS30 PROT CLR	PCS29 PROT CLR	PCS28 PROT CLR	PCS27 PROT CLR	PCS26 PROT CLR	PCS25 PROT CLR	PCS24 PROT CLR	PCS23 PROT CLR	PCS22 PROT CLR	PCS21 PROT CLR	PCS20 PROT CLR	PCS19 PROT CLR	PCS18 PROT CLR	PCS17 PROT CLR	PCS16 PROT CLR
R/WP-0														

14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCS14 PROT CLR	PCS13 PROT CLR	PCS12 PROT CLR	PCS11 PROT CLR	PCS10 PROT CLR	PCS9 PROT CLR	PCS8 PROT CLR	PCS7 PROT CLR	PCS6 PROT CLR	PCS5 PROT CLR	PCS4 PROT CLR	PCS3 PROT CLR	PCS2 PROT CLR	PCS1 PROT CLR	PCS0 PROT CLR
R/WP-0														

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 4-61. Peripheral Memory Protection Clear Register 0 (PMPROTCLR0) Field Descriptions

Bit	Name	Value	Description
31–0	PCS[31:0]PROTCLR		Peripheral memory frame protection set.
		0	Read: The peripheral memory frame[31:0] can be written to and read from in both user and privileged modes. Write: The bit is unchanged.
		1	Read: The peripheral memory frame[31:0] 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.

4.3.4 Peripheral Memory Protection Clear Register1 (PMPROTCLR1)

This register is shown in Figure 4-61 and described in Table 4-62.

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

Figure 4-61. Peripheral Memory Protection Clear Register1 (PMPROTCLR1) [offset = 0x14]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PCS63 PROT CLR	PCS62 PROT CLR	PCS61 PROT CLR	PCS60 PROT CLR	PCS59 PROT CLR	PCS58 PROT CLR	PCS57 PROT CLR	PCS56 PROT CLR	PCS55 PROT CLR	PCS54 PROT CLR	PCS53 PROT CLR	PCS52 PROT CLR	PCS51 PROT CLR	PCS50 PROT CLR	PCS49 PROT CLR	PCS48 PROT CLR
R/WP-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCS47 PROT CLR	PCS46 PROT CLR	PCS45 PROT CLR	PCS44 PROT CLR	PCS43 PROT CLR	PCS42 PROT CLR	PCS41 PROT CLR	PCS40 PROT CLR	PCS39 PROT CLR	PCS38 PROT CLR	PCS37 PROT CLR	PCS36 PROT CLR	PCS35 PROT CLR	PCS34 PROT CLR	PCS33 PROT CLR	PCS32 PROT CLR
R/WP-0															

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 4-62. Peripheral Memory Protection Clear Register1 (PMPROTCLR1) Field Descriptions

Bit	Name	Value	Description
31–0	PCS[63:32]PROTCLR		Peripheral memory frame protection clear.
		0	Read: The peripheral memory frame[63:32] can be written to and read from in both user and privileged modes. Write: The bit is unchanged.
		1	Read: The peripheral memory frame[63:32] 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.

4.3.5 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 4-63 protects the slave.

The above arrangement is true for all the peripheral select (PS0 to PS31), presented in section 4.3.6—section 4.3.12. This register holds bits for PS0 to PS7 and is shown in Figure 4-62 and described in Table 4-63.

Note:
Writes to non implemented bits have no effect and reads are 0.

Figure 4-62. Peripheral Protection Set Register 0 (PPROTSET0) [offset = 0x20]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PS7 QUAD3 PROT SET	PS7 QUAD2 PROT SET	PS7 QUAD1 PROT SET	PS7 QUAD0 PROT SET	PS6 QUAD3 PROT SET	PS6 QUAD2 PROT SET	PS6 QUAD1 PROT SET	PS6 QUAD0 PROT SET	PS5 QUAD3 PROT SET	PS5 QUAD2 PROT SET	PS5 QUAD1 PROT SET	PS5 QUAD0 PROT SET	PS4 QUAD3 PROT SET	PS4 QUAD2 PROT SET	PS4 QUAD1 PROT SET	PS4 QUAD0 PROT SET
R/WP-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS3 QUAD3 PROT SET	PS3 QUAD2 PROT SET	PS3 QUAD1 PROT SET	PS3 QUAD0 PROT SET	PS2 QUAD3 PROT SET	PS2 QUAD2 PROT SET	PS2 QUAD1 PROT SET	PS2 QUAD0 PROT SET	PS1 QUAD3 PROT SET	PS1 QUAD2 PROT SET	PS1 QUAD1 PROT SET	PS1 QUAD0 PROT SET	PS0 QUAD3 PROT SET	PS0 QUAD2 PROT SET	PS0 QUAD1 PROT SET	PS0 QUAD0 PROT SET
R/WP-0															

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 4-63. Peripheral Protection Set Register 0 (PPROTSET0)) Field Descriptions

Bit	Name	Value	Description
31–0	PCS[7:0]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 can be read in both user and privileged modes. Write: The bit in PMPROTSET0 and PMPROTCLR0 registers is set to 1.

4.3.6 Peripheral Protection Set Register 1 (PPROTSET1)

There is one bit for each quadrant for PS8 to PS15. The protection scheme is described in PPROTSET0, in section 4.3.5. This register is shown in Figure 4-63 and described in Table 4-64.

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

Figure 4-63. Peripheral Protection Set Register 1 (PPROTSET1) [offset = 0x24]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PS15 QUAD3 PROT SET	PS15 QUAD2 PROT SET	PS15 QUAD1 PROT SET	PS15 QUAD0 PROT SET	PS14 QUAD3 PROT SET	PS14 QUAD2 PROT SET	PS14 QUAD1 PROT SET	PS14 QUAD0 PROT SET	PS13 QUAD3 PROT SET	PS13 QUAD2 PROT SET	PS13 QUAD1 PROT SET	PS13 QUAD0 PROT SET	PS12 QUAD3 PROT SET	PS12 QUAD2 PROT SET	PS12 QUAD1 PROT SET	PS12 QUAD0 PROT SET
R/WP-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS11 QUAD3 PROT SET	PS11 QUAD2 PROT SET	PS11 QUAD1 PROT SET	PS11 QUAD0 PROT SET	PS10 QUAD3 PROT SET	PS10 QUAD2 PROT SET	PS10 QUAD1 PROT SET	PS10 QUAD0 PROT SET	PS9 QUAD3 PROT SET	PS9 QUAD2 PROT SET	PS9 QUAD1 PROT SET	PS9 QUAD0 PROT SET	PS8 QUAD3 PROT SET	PS8 QUAD2 PROT SET	PS8 QUAD1 PROT SET	PS8 QUAD0 PROT SET
R/WP-0															

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 4-64. Peripheral Protection Set Register 1 (PPROTSET1) Field Descriptions

Bit	Name	Value	Description
31–0	PCS[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 can be read in both user and privileged modes. Write: The bit in PMPROTSET1 and PMPROTCLR1 registers is set to 1.

4.3.7 Peripheral Protection Set Register 2 (PPROTSET2)

There is one bit for each quadrant for PS16 to PS23. The protection scheme is described in PPROTSET0, in section 4.3.5. This register is shown in Figure 4-64 and described in Table 4-65.

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

Figure 4-64. Peripheral Protection Set Register 2 (PPROTSET2) [offset = 0x28]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PS23 QUAD3 PROT SET	PS23 QUAD2 PROT SET	PS23 QUAD1 PROT SET	PS23 QUAD0 PROT SET	PS22 QUAD3 PROT SET	PS22 QUAD2 PROT SET	PS22 QUAD1 PROT SET	PS22 QUAD0 PROT SET	PS21 QUAD3 PROT SET	PS21 QUAD2 PROT SET	PS21 QUAD1 PROT SET	PS21 QUAD0 PROT SET	PS20 QUAD3 PROT SET	PS20 QUAD2 PROT SET	PS20 QUAD1 PROT SET	PS20 QUAD0 PROT SET
R/WP-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS19 QUAD3 PROT SET	PS19 QUAD2 PROT SET	PS19 QUAD1 PROT SET	PS19 QUAD0 PROT SET	PS18 QUAD3 PROT SET	PS18 QUAD2 PROT SET	PS18 QUAD1 PROT SET	PS18 QUAD0 PROT SET	PS17 QUAD3 PROT SET	PS17 QUAD2 PROT SET	PS17 QUAD1 PROT SET	PS17 QUAD0 PROT SET	PS16 QUAD3 PROT SET	PS16 QUAD2 PROT SET	PS16 QUAD1 PROT SET	PS16 QUAD0 PROT SET
R/WP-0															

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 4-65. Peripheral Protection Set Register 2 (PPROTSET2) Field Descriptions

Bit	Name	Value	Description
31–0	PCS[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 can be read in both user and privileged modes. Write: The bit in PMPROTSET2 and PMPROTCLR2 registers is set to 1.

4.3.8 Peripheral Protection Set Register 3 (PPROTSET3)

There is one bit for each quadrant for PS24 to PS31. The protection scheme is described in PPROTSET0, in section 4.3.5. This register is shown in Figure 4-65 and described in Table 4-66.

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

Figure 4-65. Peripheral Protection Set Register 3 (PPROTSET3) [offset = 0x2C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PS31 QUAD3 PROT SET	PS31 QUAD2 PROT SET	PS31 QUAD1 PROT SET	PS31 QUAD0 PROT SET	PS30 QUAD3 PROT SET	PS30 QUAD2 PROT SET	PS30 QUAD1 PROT SET	PS30 QUAD0 PROT SET	PS29 QUAD3 PROT SET	PS29 QUAD2 PROT SET	PS29 QUAD1 PROT SET	PS29 QUAD0 PROT SET	PS28 QUAD3 PROT SET	PS28 QUAD2 PROT SET	PS28 QUAD1 PROT SET	PS28 QUAD0 PROT SET
R/WP-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS27 QUAD3 PROT SET	PS27 QUAD2 PROT SET	PS27 QUAD1 PROT SET	PS27 QUAD0 PROT SET	PS26 QUAD3 PROT SET	PS26 QUAD2 PROT SET	PS26 QUAD1 PROT SET	PS26 QUAD0 PROT SET	PS25 QUAD3 PROT SET	PS25 QUAD2 PROT SET	PS25 QUAD1 PROT SET	PS25 QUAD0 PROT SET	PS24 QUAD3 PROT SET	PS24 QUAD2 PROT SET	PS24 QUAD1 PROT SET	PS24 QUAD0 PROT SET
R/WP-0															

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 4-66. Peripheral Protection Set Register 3 (PPROTSET3) Field Descriptions

Bit	Name	Value	Description
31–0	PCS[31:0]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 can be read in both user and privileged modes. Write: The bit in PMPROTSET3 and PMPROTCLR3 registers is set to 1.

4.3.9 Peripheral Protection Clear Register 0 (PPROTCLR0)

There is one bit for each quadrant for PS0 to PS7. The protection scheme is described in PPROTSET0, in section 4.3.5. This register is shown in Figure 4-66 and described in Table 4-67.

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

Figure 4-66. Peripheral Protection Clear Register 0 (PPROTCLR0) [offset = 0x40]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PS7 QUAD3 PROT CLR	PS7 QUAD2 PROT CLR	PS7 QUAD1 PROT CLR	PS7 QUAD0 PROT CLR	PS6 QUAD3 PROT CLR	PS6 QUAD2 PROT CLR	PS6 QUAD1 PROT CLR	PS6 QUAD0 PROT CLR	PS5 QUAD3 PROT CLR	PS5 QUAD2 PROT CLR	PS5 QUAD1 PROT CLR	PS5 QUAD0 PROT CLR	PS4 QUAD3 PROT CLR	PS4 QUAD2 PROT CLR	PS4 QUAD1 PROT CLR	PS4 QUAD0 PROT CLR
R/WP-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS3 QUAD3 PROT CLR	PS3 QUAD2 PROT CLR	PS3 QUAD1 PROT CLR	PS3 QUAD0 PROT CLR	PS2 QUAD3 PROT CLR	PS2 QUAD2 PROT CLR	PS2 QUAD1 PROT CLR	PS2 QUAD0 PROT CLR	PS1 QUAD3 PROT CLR	PS1 QUAD2 PROT CLR	PS1 QUAD1 PROT CLR	PS1 QUAD0 PROT CLR	PS0 QUAD3 PROT CLR	PS0 QUAD2 PROT CLR	PS0 QUAD1 PROT CLR	PS0 QUAD0 PROT CLR
R/WP-0															

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 4-67. Peripheral Protection Clear Register 0 (PPROTCLR0) Field Descriptions

Bit	Name	Value	Description
31–0	PCS[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 can be read in both user and privileged modes. Write: The bit in PMPROTSET0 and PMPROTCLR0 registers is clear to 0.

4.3.10 Peripheral Protection Clear Register 1 (PPROTCLR1)

There is one bit for each quadrant for PS8 to PS15. The protection scheme is described in PPROTSET0, in section 4.3.5. This register is shown in Figure 4-67 and described in Table 4-68.

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

Figure 4-67. Peripheral Protection Clear Register 1 (PPROTCLR1) [offset = 0x44]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PS15 QUAD 3 PROT CLR	PS15 QUAD 2 PROT CLR	PS15 QUAD 1 PROT CLR	PS15 QUAD 0 PROT CLR	PS14 QUAD 3 PROT CLR	PS14 QUAD 2 PROT CLR	PS14 QUAD 1 PROT CLR	PS14 QUAD 0 PROT CLR	PS13 QUAD 3 PROT CLR	PS13 QUAD 2 PROT CLR	PS13 QUAD 1 PROT CLR	PS13 QUAD 0 PROT CLR	PS12 QUAD 3 PROT CLR	PS12 QUAD 2 PROT CLR	PS12 QUAD 1 PROT CLR	PS12 QUAD 0 PROT CLR
R/WP-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS11 QUAD 3 PROT CLR	PS11 QUAD 2 PROT CLR	PS11 QUAD 1 PROT CLR	PS11 QUAD 0 PROT CLR	PS10 QUAD 3 PROT CLR	PS10 QUAD 2 PROT CLR	PS10 QUAD 1 PROT CLR	PS10 QUAD 0 PROT CLR	PS9 QUAD 3 PROT CLR	PS9 QUAD 2 PROT CLR	PS9 QUAD 1 PROT CLR	PS9 QUAD 0 PROT CLR	PS8 QUAD 3 PROT CLR	PS8 QUAD 2 PROT CLR	PS8 QUAD 1 PROT CLR	PS8 QUAD 0 PROT CLR
R/WP-0															

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 4-68. Peripheral Protection Clear Register 1 (PPROTCLR1) Field Descriptions

Bit	Name	Value	Description
31–0	PCS[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 can be read in both user and privileged modes. Write: The bit in PMPROTSET1 and PMPROTCLR1 registers is clear to 0.

4.3.11 Peripheral Protection Clear Register 2 (PPROTCLR2)

There is one bit for each quadrant for PS16 to PS23. The protection scheme is described in PPROTSET0, in section 4.3.5. This register is shown in Figure 4-68 and described in Table 4-69.

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

Figure 4-68. Peripheral Protection Clear Register 2 (PPROTCLR2) [offset = 0x48]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PS23 QUAD3 PROT CLR	PS23 QUAD2 PROT CLR	PS23 QUAD1 PROT CLR	PS23 QUAD0 PROT CLR	PS22 QUAD3 PROT CLR	PS22 QUAD2 PROT CLR	PS22 QUAD1 PROT CLR	PS22 QUAD0 PROT CLR	PS21 QUAD3 PROT CLR	PS21 QUAD2 PROT CLR	PS21 QUAD1 PROT CLR	PS21 QUAD0 PROT CLR	PS20 QUAD3 PROT CLR	PS20 QUAD2 PROT CLR	PS20 QUAD1 PROT CLR	PS20 QUAD0 PROT CLR
R/WP-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS19 QUAD3 PROT CLR	PS19 QUAD2 PROT CLR	PS19 QUAD1 PROT CLR	PS19 QUAD0 PROT CLR	PS18 QUAD3 PROT CLR	PS18 QUAD2 PROT CLR	PS18 QUAD1 PROT CLR	PS18 QUAD0 PROT CLR	PS17 QUAD3 PROT CLR	PS17 QUAD2 PROT CLR	PS17 QUAD1 PROT CLR	PS17 QUAD0 PROT CLR	PS16 QUAD3 PROT CLR	PS16 QUAD2 PROT CLR	PS16 QUAD1 PROT CLR	PS16 QUAD0 PROT CLR
R/WP-0															

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 4-69. Peripheral Protection Clear Register 2 (PPROTCLR2) Field Descriptions

Bit	Name	Value	Description
31–0	PCS[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 can be read in both user and privileged modes. Write: The bit in PMPROTSET2 and PMPROTCLR2 registers is clear to 0.

4.3.12 Peripheral Protection Clear Register 3 (PPROTCLR3)

There is one bit for each quadrant for PS24 to PS31. The protection scheme is described in PPROTSET0, in section 4.3.5. This register is shown in Figure 4-69 and described in Table 4-70.

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

Figure 4-69. Peripheral Protection Clear Register 3 (PPROTCLR3) [offset = 0x4C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PS31 QUAD3 PROT CLR	PS31 QUAD2 PROT CLR	PS31 QUAD1 PROT CLR	PS31 QUAD0 PROT CLR	PS30 QUAD3 PROT CLR	PS30 QUAD2 PROT CLR	PS30 QUAD1 PROT CLR	PS30 QUAD0 PROT CLR	PS29 QUAD3 PROT CLR	PS29 QUAD2 PROT CLR	PS29 QUAD1 PROT CLR	PS29 QUAD0 PROT CLR	PS28 QUAD3 PROT CLR	PS28 QUAD2 PROT CLR	PS28 QUAD1 PROT CLR	PS28 QUAD0 PROT CLR
R/WP-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS27 QUAD3 PROT CLR	PS27 QUAD2 PROT CLR	PS27 QUAD1 PROT CLR	PS27 QUAD0 PROT CLR	PS26 QUAD3 PROT CLR	PS26 QUAD2 PROT CLR	PS26 QUAD1 PROT CLR	PS26 QUAD0 PROT CLR	PS25 QUAD3 PROT CLR	PS25 QUAD2 PROT CLR	PS25 QUAD1 PROT CLR	PS25 QUAD0 PROT CLR	PS24 QUAD3 PROT CLR	PS24 QUAD2 PROT CLR	PS24 QUAD1 PROT CLR	PS24 QUAD0 PROT CLR
R/WP-0															

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 4-70. Peripheral Protection Clear Register 3 (PPROTCLR3) Field Descriptions

Bit	Name	Value	Description
31–0	PCS[31:0]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 can be read in both user and privileged modes. Write: The bit in PMPROTSET3 and PMPROTCLR3 registers is clear to 0.

4.3.13 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.

Note:
Only those bits that have a slave at the corresponding bit position are implemented. Writes to non implemented bits have no effect and reads are 0. For non-independent slaves in the PCS frame, the clocks are expected to be controlled by the sharing peripheral.

This register is shown in Figure 4-70 and described in Table 4-71.

Figure 4-70. Peripheral Memory Power-Down Set Register 0 (PCSPWRDWNSET0) [offset = 0x60]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PCS31 PWR DWN SET	PCS30 PWR DWN SET	PCS29 PWR DWN SET	PCS28 PWR DWN SET	PCS27 PWR DWN SET	PCS26 PWR DWN SET	PCS25 PWR DWN SET	PCS24 PWR DWN SET	PCS23 PWR DWN SET	PCS22 PWR DWN SET	PCS21 PWR DWN SET	PCS20 PWR DWN SET	PCS19 PWR DWN SET	PCS18 PWR DWN SET	PCS17 PWR DWN SET	PCS16 PWR DWN SET
R/WP-1															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCS15 PWR DWN SET	PCS14 PWR DWN SET	PCS13 PWR DWN SET	PCS12 PWR DWN SET	PCS11 PWR DWN SET	PCS10 PWR DWN SET	PCS9 PWR DWN SET	PCS8 PWR DWN SET	PCS7 PWR DWN SET	PCS6 PWR DWN SET	PCS5 PWR DWN SET	PCS4 PWR DWN SET	PCS3 PWR DWN SET	PCS2 PWR DWN SET	PCS1 PWR DWN SET	PCS0 PWR DWN SET
R/WP-1															

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 4-71. Peripheral Memory Power-Down Set Register 0 (PCSPWRDWNSET0) Field Descriptions

Bit	Name	Value	Description
31–0	PCS[31:0] PWRDNSET		Peripheral memory clock power-down set.
		0	Read: The peripheral memory clock[31:0] is not to be powered down. Write: The bit is unchanged.
		1	Read: The peripheral memory clock[31:0] needs to be powered down. Write: The corresponding bit in the PCSPWRDWNSET0 and PCSPWRDWNCLR0 registers is set to 1.

4.3.14 Peripheral Memory Power-Down Set Register1 (PCSPWRDWNSET1)

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

This register is shown in Figure 4-71 and described in Table 4-74.

Figure 4-71. Peripheral Memory Power-Down Set Register1 (PCSPWRDWNSET1) [offset = 0x64]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PCS63 PWR DWN SET	PCS62 PWR DWN SET	PCS61 PWR DWN SET	PCS60 PWR DWN SET	PCS59 PWR DWN SET	PCS58 PWR DWN SET	PCS57 PWR DWN SET	PCS56 PWR DWN SET	PCS55 PWR DWN SET	PCS54 PWR DWN SET	PCS53 PWR DWN SET	PCS52 PWR DWN SET	PCS51 PWR DWN SET	PCS50 PWR DWN SET	PCS49 PWR DWN SET	PCS48 PWR DWN SET
R/WP-1															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCS47 PWR DWN SET	PCS46 PWR DWN SET	PCS45 PWR DWN SET	PCS44 PWR DWN SET	PCS43 PWR DWN SET	PCS42 PWR DWN SET	PCS41 PWR DWN SET	PCS40 PWR DWN SET	PCS39 PWR DWN SET	PCS38 PWR DWN SET	PCS37 PWR DWN SET	PCS36 PWR DWN SET	PCS35 PWR DWN SET	PCS34 PWR DWN SET	PCS33 PWR DWN SET	PCS32 PWR DWN SET
R/WP-1															

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 4-72. Peripheral Memory Power-Down Set Register1 (PCSPWRDWNSET1) Field Descriptions

Bit	Name	Value	Description
31–0	PCS[63:32] PWRDNSET		Peripheral memory clock power-down set.
		0	Read: The peripheral memory clock[63:32] is not to be powered down. Write: The bit is unchanged.
		1	Read: The peripheral memory clock[63:32] needs to be powered down. Write: The corresponding bit in the PCSPWRDWNSET1 and PCSPWRDWNCLR1 registers is set to 1.

4.3.15 Peripheral Memory Power-Down Clear Register 0 (PCSPWRDWNCLR0)

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

This register is shown in Figure 4-72 and described in Table 4-73.

Figure 4-72. Peripheral Memory Power-Down Clear Register 0 (PCSPWRDWNCLR0) [offset = 0x70]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PCS31 PWR DWN CLR	PCS30 PWR DWN CLR	PCS29 PWR DWN CLR	PCS28 PWR DWN CLR	PCS27 PWR DWN CLR	PCS26 PWR DWN CLR	PCS25 PWR DWN CLR	PCS24 PWR DWN CLR	PCS23 PWR DWN CLR	PCS22 PWR DWN CLR	PCS21 PWR DWN CLR	PCS20 PWR DWN CLR	PCS19 PWR DWN CLR	PCS18 PWR DWN CLR	PCS17 PWR DWN CLR	PCS16 PWR DWN CLR
R/WP-1															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCS15 PWR DWN CLR	PCS14 PWR DWN CLR	PCS13 PWR DWN CLR	PCS12 PWR DWN CLR	PCS11 PWR DWN CLR	PCS10 PWR DWN CLR	PCS9 PWR DWN CLR	PCS8 PWR DWN CLR	PCS7 PWR DWN CLR	PCS6 PWR DWN CLR	PCS5 PWR DWN CLR	PCS4 PWR DWN CLR	PCS3 PWR DWN CLR	PCS2 PWR DWN CLR	PCS1 PWR DWN CLR	PCS0 PWR DWN CLR
R/WP-1															

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 4-73. Peripheral Memory Power-Down Clear Register 0 (PCSPWRDWNCLR0) Field Descriptions

Bit	Name	Value	Description
31–0	PCS[31:0] PWRDNCLR0		Peripheral memory clock power-down clear.
		0	Read: The peripheral memory clock[31:0] is not to be powered down. Write: The bit is unchanged.
		1	Read: The peripheral memory clock[31:0] needs to be powered down. Write: The corresponding bit in the PCSPWRDWNSET0 and PCSPWRDWNCLR0 registers is cleared to 0.

4.3.16 Peripheral Memory Power-Down Clear Register1 (PCSPWRDWNCLR1)

Note:
Only those bits that have a slave at the corresponding bit position are implemented. Therefore, the size of this register is device-dependent. Writes to non implemented bits have no effect and reads are 0.

This register is shown in Figure 4-73 and described in Table 4-74.

Figure 4-73. Peripheral Memory Power-Down Clear Register1 (PCSPWRDWNCLR1) [offset = 0x74]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PCS63 PWR DWN CLR	PCS62 PWR DWN CLR	PCS61 PWR DWN CLR	PCS60 PWR DWN CLR	PCS59 PWR DWN CLR	PCS58 PWR DWN CLR	PCS57 PWR DWN CLR	PCS56 PWR DWN CLR	PCS55 PWR DWN CLR	PCS54 PWR DWN CLR	PCS53 PWR DWN CLR	PCS52 PWR DWN CLR	PCS51 PWR DWN CLR	PCS50 PWR DWN CLR	PCS49 PWR DWN CLR	PCS48 PWR DWN CLR
R/WP-1															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCS47 PWR DWN CLR	PCS46 PWR DWN CLR	PCS45 PWR DWN CLR	PCS44 PWR DWN CLR	PCS43 PWR DWN CLR	PCS42 PWR DWN CLR	PCS41 PWR DWN CLR	PCS40 PWR DWN CLR	PCS39 PWR DWN CLR	PCS38 PWR DWN CLR	PCS37 PWR DWN CLR	PCS36 PWR DWN CLR	PCS35 PWR DWN CLR	PCS34 PWR DWN CLR	PCS33 PWR DWN CLR	PCS32 PWR DWN CLR
R/WP-1															

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 4-74. Peripheral Memory Power-Down Set Register1 (PCSPWRDWNCLR1) Field Descriptions

Bit	Name	Value	Description
31–0	PCS[63:32] PWRDNSET		Peripheral memory clock power-down clear.
		0	Read: The peripheral memory clock[63:32] is not to be powered down. Write: The bit is unchanged.
		1	Read: The peripheral memory clock[63:32] needs to be powered down. Write: The corresponding bit in the PCSPWRDWNSET1 and PCSPWRDWNCLR1 registers is cleared to 0.

4.3.17 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 4.3.5.

This arrangement is the same for bits of PS8 to PS31, presented in section 4.3.18—section 4.3.24. This register holds bits for PS0 to PS7. This register is shown in Figure 4-74 and described in Table 4-75.

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

Figure 4-74. Peripheral Power-Down Set Register 0 (PSPWRDWNSET0) [offset = 0x80]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PS7 QUAD 3 PWR DWN SET	PS7 QUAD 2 PWR DWN SET	PS7 QUAD 1 PWR DWN SET	PS7 QUAD 0 PWR DWN SET	PS6 QUAD 3 PWR DWN SET	PS6 QUAD 2 PWR DWN SET	PS6 QUAD 1 PWR DWN SET	PS6 QUAD 0 PWR DWN SET	PS5 QUAD 3 PWR DWN SET	PS5 QUAD 2 PWR DWN SET	PS5 QUAD 1 PWR DWN SET	PS5 QUAD 0 PWR DWN SET	PS4 QUAD 3 PWR DWN SET	PS4 QUAD 2 PWR DWN SET	PS4 QUAD 1 PWR DWN SET	PS4 QUAD 0 PWR DWN SET
R/WP-1															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS3 QUAD 3 PWR DWN SET	PS3 QUAD 2 PWR DWN SET	PS3 QUAD 1 PWR DWN SET	PS3 QUAD 0 PWR DWN SET	PS2 QUAD 3 PWR DWN SET	PS2 QUAD 2 PWR DWN SET	PS2 QUAD 1 PWR DWN SET	PS2 QUAD 0 PWR DWN SET	PS1 QUAD 3 PWR DWN SET	PS1 QUAD 2 PWR DWN SET	PS1 QUAD 1 PWR DWN SET	PS1 QUAD 0 PWR DWN SET	PS0 QUAD 3 PWR DWN SET	PS0 QUAD 2 PWR DWN SET	PS0 QUAD 1 PWR DWN SET	PS0 QUAD 0 PWR DWN SET
R/WP-1															

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 4-75. Peripheral Power-Down Set Register 0 (PSPWRDWNSET0) Field Descriptions

Bit	Name	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 needs to be powered up. Write: The bit is unchanged.
		1	Read: The clock to the peripheral select quadrant needs to be powered down. Write: The corresponding bit in PSPWRDWNSET0 and PSPWRDWNCLR0 registers is set to 1

4.3.18 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 4.3.17. This register is shown in Figure 4-75 and described in Table 4-76.

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

Figure 4-75. Peripheral Power-Down Set Register 1 (PSPWRDWNSET1) [offset = 0x84]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PS15 QUAD3 PWR DWN SET	PS15 QUAD2 PWR DWN SET	PS15 QUAD1 PWR DWN SET	PS15 QUAD0 PWR DWN SET	PS14 QUAD3 PWR DWN SET	PS14 QUAD2 PWR DWN SET	PS14 QUAD1 PWR DWN SET	PS14 QUAD0 PWR DWN SET	PS13 QUAD3 PWR DWN SET	PS13 QUAD2 PWR DWN SET	PS13 QUAD1 PWR DWN SET	PS13 QUAD0 PWR DWN SET	PS12 QUAD3 PWR DWN SET	PS12 QUAD2 PWR DWN SET	PS12 QUAD1 PWR DWN SET	PS12 QUAD0 PWR DWN SET
R/WP-1															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS11 QUAD3 PWR DWN SET	PS11 QUAD2 PWR DWN SET	PS11 QUAD1 PWR DWN SET	PS11 QUAD0 PWR DWN SET	PS10 QUAD3 PWR DWN SET	PS10 QUAD2 PWR DWN SET	PS10 QUAD1 PWR DWN SET	PS10 QUAD0 PWR DWN SET	PS9 QUAD3 PWR DWN SET	PS9 QUAD2 PWR DWN SET	PS9 QUAD1 PWR DWN SET	PS9 QUAD0 PWR DWN SET	PS8 QUAD3 PWR DWN SET	PS8 QUAD2 PWR DWN SET	PS8 QUAD1 PWR DWN SET	PS8 QUAD0 PWR DWN SET
R/WP-1															

R/WP = Read in all modes, any time, write in privileged mode only, -n = Value after reset

.

Table 4-76. Peripheral Power-Down Set Register 1 (PSPWRDWNSET1) Field Descriptions

Bit	Name	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 needs to be powered up. Write: The bit is unchanged.
		1	Read: The clock to the peripheral select quadrant needs to be powered down. Write: The corresponding bit in PSPWRDWNSET1 and PSPWRDWNCLR1 registers is set to 1

4.3.19 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 4.3.17. This register is shown in Figure 4-69 and described in Table 4-70.

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

Figure 4-76. Peripheral Power-Down Set Register 2 (PSPWRDWNSET2) [offset = 0x88]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PS23 QUAD3 PWR DWN SET	PS23 QUAD2 PWR DWN SET	PS23 QUAD1 PWR DWN SET	PS23 QUAD0 PWR DWN SET	PS22 QUAD3 PWR DWN SET	PS22 QUAD2 PWR DWN SET	PS22 QUAD1 PWR DWN SET	PS22 QUAD0 PWR DWN SET	PS21 QUAD3 PWR DWN SET	PS21 QUAD2 PWR DWN SET	PS21 QUAD1 PWR DWN SET	PS21 QUAD0 PWR DWN SET	PS20 QUAD3 PWR DWN SET	PS20 QUAD2 PWR DWN SET	PS20 QUAD1 PWR DWN SET	PS20 QUAD0 PWR DWN SET
R/WP-1															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS19 QUAD3 PWR DWN SET	PS19 QUAD2 PWR DWN SET	PS19 QUAD1 PWR DWN SET	PS19 QUAD0 PWR DWN SET	PS18 QUAD3 PWR DWN SET	PS18 QUAD2 PWR DWN SET	PS18 QUAD1 PWR DWN SET	PS18 QUAD0 PWR DWN SET	PS17 QUAD3 PWR DWN SET	PS17 QUAD2 PWR DWN SET	PS17 QUAD1 PWR DWN SET	PS17 QUAD0 PWR DWN SET	PS16 QUAD3 PWR DWN SET	PS16 QUAD2 PWR DWN SET	PS16 QUAD1 PWR DWN SET	PS16 QUAD0 PWR DWN SET
R/WP-1															

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 4-77. Peripheral Power-Down Set Register 2 (PSPWRDWNSET2) Field Descriptions

Bit	Name	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 needs to be powered up. Write: The bit is unchanged.
		1	Read: The clock to the peripheral select quadrant needs to be powered down. Write: The corresponding bit in PSPWRDWNSET2 and PSPWRDWNCLR2 registers is set to 1

4.3.20 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 4.3.17. This register is shown in Figure 4-77 and described in Table 4-78.

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

Figure 4-77. Peripheral Power-Down Set Register 3 (PSPWRDWNSET3) [offset = 0x8C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PS31 QUAD3 PWR DWN SET	PS31 QUAD2 PWR DWN SET	PS31 QUAD1 PWR DWN SET	PS31 QUAD0 PWR DWN SET	PS30 QUAD3 PWR DWN SET	PS30 QUAD2 PWR DWN SET	PS30 QUAD1 PWR DWN SET	PS30 QUAD0 PWR DWN SET	PS29 QUAD3 PWR DWN SET	PS29 QUAD2 PWR DWN SET	PS29 QUAD1 PWR DWN SET	PS29 QUAD0 PWR DWN SET	PS28 QUAD3 PWR DWN SET	PS28 QUAD2 PWR DWN SET	PS28 QUAD1 PWR DWN SET	PS28 QUAD0 PWR DWN SET
R/WP-1															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS27 QUAD3 PWR DWN SET	PS27 QUAD2 PWR DWN SET	PS27 QUAD1 PWR DWN SET	PS27 QUAD0 PWR DWN SET	PS26 QUAD3 PWR DWN SET	PS26 QUAD2 PWR DWN SET	PS26 QUAD1 PWR DWN SET	PS26 QUAD0 PWR DWN SET	PS25 QUAD3 PWR DWN SET	PS25 QUAD2 PWR DWN SET	PS25 QUAD1 PWR DWN SET	PS25 QUAD0 PWR DWN SET	PS24 QUAD3 PWR DWN SET	PS24 QUAD2 PWR DWN SET	PS24 QUAD1 PWR DWN SET	PS24 QUAD0 PWR DWN SET
R/WP-1															

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 4-78. Peripheral Power-Down Set Register 3 (PSPWRDWNSET3) Field Descriptions

Bit	Name	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 needs to be powered up. Write: The bit is unchanged.
		1	Read: The clock to the peripheral select quadrant needs to be powered down. Write: The corresponding bit in PSPWRDWNSET3 and PSPWRDWNCLR3 registers is set to 1

4.3.21 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 4.3.17. This register is shown in Figure 4-78 and described in Table 4-79.

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

Figure 4-78. Peripheral Power-Down Clear Register 0 (PSPWRDWNCLR0) [offset = 0xA0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PS7 QUAD3 PWR DWN CLR	PS7 QUAD2 PWR DWN CLR	PS7 QUAD1 PWR DWN CLR	PS7 QUAD0 PWR DWN CLR	PS6 QUAD3 PWR DWN CLR	PS6 QUAD2 PWR DWN CLR	PS6 QUAD1 PWR DWN CLR	PS6 QUAD0 PWR DWN CLR	PS5 QUAD3 PWR DWN CLR	PS5 QUAD2 PWR DWN CLR	PS5 QUAD1 PWR DWN CLR	PS5 QUAD0 PWR DWN CLR	PS4 QUAD3 PWR DWN CLR	PS4 QUAD2 PWR DWN CLR	PS4 QUAD1 PWR DWN CLR	PS4 QUAD0 PWR DWN CLR
R/WP-1															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS3 QUAD3 PWR DWN CLR	PS3 QUAD2 PWR DWN CLR	PS3 QUAD1 PWR DWN CLR	PS3 QUAD0 PWR DWN CLR	PS2 QUAD3 PWR DWN CLR	PS2 QUAD2 PWR DWN CLR	PS2 QUAD1 PWR DWN CLR	PS2 QUAD0 PWR DWN CLR	PS1 QUAD3 PWR DWN CLR	PS1 QUAD2 PWR DWN CLR	PS1 QUAD1 PWR DWN CLR	PS1 QUAD0 PWR DWN CLR	PS0 QUAD3 PWR DWN CLR	PS0 QUAD2 PWR DWN CLR	PS0 QUAD1 PWR DWN CLR	PS0 QUAD0 PWR DWN CLR
R/WP-1															

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 4-79. Peripheral Power-Down Clear Register 0 (PSPWRDWNCLR0) Field Descriptions

Bit	Name	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 needs to be powered up. Write: The bit is unchanged.
		1	Read: The clock to the peripheral select quadrant needs to be powered down. Write: The corresponding bit in PSPWRDWNSET0 and PSPWRDWNCLR0 registers is cleared to 0.

4.3.22 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 4.3.17. This register is shown in Figure 4-79 and described in Table 4-80.

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

Figure 4-79. Peripheral Power-Down Clear Register 1 (PSPWRDWNCLR1) [offset = 0xA4]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PS15 QUAD3 PWR DWN CLR	PS15 QUAD2 PWR DWN CLR	PS15 QUAD1 PWR DWN CLR	PS15 QUAD0 PWR DWN CLR	PS14 QUAD3 PWR DWN CLR	PS14 QUAD2 PWR DWN CLR	PS14 QUAD1 PWR DWN CLR	PS14 QUAD0 PWR DWN CLR	PS13 QUAD3 PWR DWN CLR	PS13 QUAD2 PWR DWN CLR	PS13 QUAD1 PWR DWN CLR	PS13 QUAD0 PWR DWN CLR	PS12 QUAD3 PWR DWN CLR	PS12 QUAD2 PWR DWN CLR	PS12 QUAD1 PWR DWN CLR	PS12 QUAD0 PWR DWN CLR
R/WP-1															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS11 QUAD3 PWR DWN CLR	PS11 QUAD2 PWR DWN CLR	PS11 QUAD1 PWR DWN CLR	PS11 QUAD0 PWR DWN CLR	PS10 QUAD3 PWR DWN CLR	PS10 QUAD2 PWR DWN CLR	PS10 QUAD1 PWR DWN CLR	PS10 QUAD0 PWR DWN CLR	PS9 QUAD3 PWR DWN CLR	PS9 QUAD2 PWR DWN CLR	PS9 QUAD1 PWR DWN CLR	PS9 QUAD0 PWR DWN CLR	PS8 QUAD3 PWR DWN CLR	PS8 QUAD2 PWR DWN CLR	PS8 QUAD1 PWR DWN CLR	PS8 QUAD0 PWR DWN CLR
R/WP-1															

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

Table 4-80. Peripheral Power-Down Clear Register 1 (PSPWRDWNCLR1) Field Descriptions

Bit	Name	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 needs to be powered up. Write: The bit is unchanged.
		1	Read: The clock to the peripheral select quadrant needs to be powered down. Write: The corresponding bit in PSPWRDWNSET0 and PSPWRDWNCLR0 registers is cleared to 0.

4.3.23 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 4.3.17. This register is shown in Figure 4-80 and described in Table 4-81.

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

Figure 4-80. Peripheral Power-Down Clear Register 2 (PSPWRDWNCLR2) [offset = 0xA8]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PS23 QUAD3 PWR DWN CLR	PS23 QUAD2 PWR DWN CLR	PS23 QUAD1 PWR DWN CLR	PS23 QUAD0 PWR DWN CLR	PS22 QUAD3 PWR DWN CLR	PS22 QUAD2 PWR DWN CLR	PS22 QUAD1 PWR DWN CLR	PS22 QUAD0 PWR DWN CLR	PS21 QUAD3 PWR DWN CLR	PS21 QUAD2 PWR DWN CLR	PS21 QUAD1 PWR DWN CLR	PS21 QUAD0 PWR DWN CLR	PS20 QUAD3 PWR DWN CLR	PS20 QUAD2 PWR DWN CLR	PS20 QUAD1 PWR DWN CLR	PS20 QUAD0 PWR DWN CLR
R/WP-1															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS19 QUAD3 PWR DWN CLR	PS19 QUAD2 PWR DWN CLR	PS19 QUAD1 PWR DWN CLR	PS19 QUAD0 PWR DWN CLR	PS18 QUAD3 PWR DWN CLR	PS18 QUAD2 PWR DWN CLR	PS18 QUAD1 PWR DWN CLR	PS18 QUAD0 PWR DWN CLR	PS17 QUAD3 PWR DWN CLR	PS17 QUAD2 PWR DWN CLR	PS17 QUAD1 PWR DWN CLR	PS17 QUAD0 PWR DWN CLR	PS16 QUAD3 PWR DWN CLR	PS16 QUAD2 PWR DWN CLR	PS16 QUAD1 PWR DWN CLR	PS16 QUAD0 PWR DWN CLR
R/WP-1															

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

Table 4-81. Peripheral Power-Down Clear Register 2 (PSPWRDWNCLR2) Field Descriptions

Bit	Name	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 needs to be powered up. Write: The bit is unchanged.
		1	Read: The clock to the peripheral select quadrant needs to be powered down. Write: The corresponding bit in PSPWRDWNSET0 and PSPWRDWNCLR0 registers is cleared to 0.

4.3.24 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 4.3.17. This register is shown in Figure 4-81 and described in Table 4-82.

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

Figure 4-81. Peripheral Power-Down Clear Register 3 (PSPWRDWNCLR) [offset = 0xAC]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PS31 QUAD3 PWR DWN CLR	PS31 QUAD2 PWR DWN CLR	PS31 QUAD1 PWR DWN CLR	PS31 QUAD0 PWR DWN CLR	PS30 QUAD3 PWR DWN CLR	PS30 QUAD2 PWR DWN CLR	PS30 QUAD1 PWR DWN CLR	PS30 QUAD0 PWR DWN CLR	PS29 QUAD3 PWR DWN CLR	PS29 QUAD2 PWR DWN CLR	PS29 QUAD1 PWR DWN CLR	PS29 QUAD0 PWR DWN CLR	PS28 QUAD3 PWR DWN CLR	PS28 QUAD2 PWR DWN CLR	PS28 QUAD1 PWR DWN CLR	PS28 QUAD0 PWR DWN CLR
R/WP-1															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS27 QUAD3 PWR DWN CLR	PS27 QUAD2 PWR DWN CLR	PS27 QUAD1 PWR DWN CLR	PS27 QUAD0 PWR DWN CLR	PS26 QUAD3 PWR DWN CLR	PS26 QUAD2 PWR DWN CLR	PS26 QUAD1 PWR DWN CLR	PS26 QUAD0 PWR DWN CLR	PS25 QUAD3 PWR DWN CLR	PS25 QUAD2 PWR DWN CLR	PS25 QUAD1 PWR DWN CLR	PS25 QUAD0 PWR DWN CLR	PS24 QUAD3 PWR DWN CLR	PS24 QUAD2 PWR DWN CLR	PS24 QUAD1 PWR DWN CLR	PS24 QUAD0 PWR DWN CLR
R/WP-1															

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

Table 4-82. Peripheral Power-Down Clear Register 3 (PSPWRDWNCLR3) Field Descriptions

Bit	Name	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 needs to be powered up. Write: The bit is unchanged.
		1	Read: The clock to the peripheral select quadrant needs to be powered down. Write: The corresponding bit in PSPWRDWNSET0 and PSPWRDWNCLR0 registers is cleared to 0.

4.4 System Control Registers Secondary Frame(SYS2)

The start address of the second system module frame is 0xFFFF E100 and the end address is 0xFFFF E1FF. The registers support 32-, 16-, and 8-bit writes.

Figure 4-82 contains a summary figure of the system control registers.

Figure 4-82. Frame 2 System Control Registers Summary

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
0x04 Reserved	Reserved															
	Reserved															
0x08 STCLKDIV Page 175	Reserved								CLKDIV				Reserved			
	Reserved															
0x0C - 0x20 Reserved	Reserved															
	Reserved															
0x24 ECPCNTRL1 Page 176	Reserved								ECPS-SEL	ECP-COS	Reserved					
	ECPDIV[15:0]															
0x28 ECPCNTRL2 Page 177	ECP2_KEY[3:0]				Reserved				ECP2SSEL	ECP2COS	Reserved					
	ECP2DIV[15:0]															

4.4.1 CPU Logic BIST Clock Divider (STCLKDIV)

This register is shown in Figure 4-83 and described in Table 4-83.

Figure 4-83. CPU Logic BIST Clock Prescaler (STCLKDIV) [offset = 0x08]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
					CLKDIV			Reserved							
R-0					RWP-000			R-0							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															
R-0															

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset; D = device specific

.

Table 4-83. CPU Logic BIST Clock Prescaler (STCLKDIV) Field Descriptions

Bit	Name	Value	Description
31–27	Reserved		Reads are zero and writes have no effect
26-24	CLKDIV		Clock divider/prescaler for CPU clock during logic BIST
		000	div by 1
		001	div by 2
		010	div by 3
		011	div by 4
		...	...
		111	div by 8
23-0	Reserved		Reads are zero and writes have no effect

1. The node must be transmitter
2. The node must transmit a dominant bit
3. The dominant bit must be sampled back as recessive
4. A recessive to dominant edge must be detected after the sample point

4.4.2 ECP Control Register (ECPCNTL1)

This register is shown in Figure 4-84 and described in Table 4-84. This reflects the ECPCNTL register present in the SYS1 frame located at address 0xFFFFF_FFD4.

Note:
This register is only available on some configurations of TMS470M series that supports ECPCLK2. Please check the device specific datasheet for applicability to the specific part number being used.

Figure 4-84. ECP Control Register (ECPCNTL1) [offset = 24h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved							ECP-SSEL	ECP COS	Reserved						
R-0							R/W-0	R/W-0	R-0						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ECPDIV[15:0]															
R/W-0															

R = Read; WP = Write in privileged mode only; *-n* = Value after reset

.

Table 4-84. ECP Control Register (ECPCNTL1) Field Descriptions

Bit	Name	Value	Description
31–25	Reserved		Reads return zero and writes have no effect.
24	ECPSSEL		ECP Source Clock Select for ECP module.
		1	ECP's source clock is oscillator.
		0	ECP's source clock is VCLK.
23	ECPCOS		ECP continue on suspend.
		0	ECPCLK output is disabled in suspend mode. ECPCLK output will be shut off and will not be seen on the I/O pin of the device.
		1	ECPCLK output is enabled in suspend mode. ECPCLK output will be seen on the I/O pin of the device.
22–16	Reserved		Reads return zero and writes have no effect.
15–0	ECPDIV(15–0)	0–FFFF	ECP divider value. The value of ECPDIV bits determine the external clock (ECP clock) frequency as a ratio of VBUS clock (VCLK) as $\text{ECPCLK} = \frac{\text{OSCIN or VCLK}}{(\text{ECPDIV} + 1)}$ **Note: The device does not guarantee ECPCLK beyond 20 MHz for (EMI) purposes.**

4.4.3 ECP Control Register (ECPCNTL2)

This register is shown in Figure 4-85 and described in Table 4-85.

Note:
This register controls ECPCLK2. ECPCLK2 is muxed with HET15. This feature/ register is only available on some configurations of TMS470M series. Please check the device specific datasheet for applicability to the specific part number being used.

Figure 4-85. ECP Control Register (ECPCNTL2) [offset = 28h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ECP2_KEY[3:0]				Reserved			ECP2-SSEL	ECP2 COS	Reserved						
R/WP-5				R-0			R/W-0	R/W-0	R-0						
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ECP2DIV[5:0]															
R/W-0															

R = Read; WP = Write in privileged mode only; *-n* = Value after reset

.

Table 4-85. ECP Control Register (ECPCNTL[3:1]) Field Descriptions

Bit	Name	Value	Description
31-28	ECP2_KEY[3:0]		Enable ECP CLK2 logic.
		0x1010	Clock functionality of ECP clock2 is enabled
		Others	Clock functionality of ECP clock2 is disabled. On reset, these bits are set to 0x5.
27–25	Reserved		Reads return zero and writes have no effect.
24	ECP2SSEL		ECP Source Clock2 Select for ECP2 module.
		1	ECP's source clock2 is oscillator.
		0	ECP's source clock2 is VCLK.
			NOTE: This field shall be exactly the same as ECPCNTL1 since both ECLKs must select the same clock source.
23	ECP2COS		ECP2 continue on suspend.
		0	ECPCLK2 output is disabled in suspend mode. ECPCLK2 output will be shut off and will not be seen on the I/O pin of the device.
		1	ECPCLK2 output is enabled in suspend mode. ECPCLK2 output will be seen on the I/O pin of the device.
22–16	Reserved		Reads return zero and writes have no effect.

Table 4-85. ECP Control Register (ECPCNTL[3:1]) Field Descriptions (Continued)

Bit	Name	Value	Description
15–0	ECP2DIV(15–0)	0–FFFF	ECP2 divider value. The value of ECP2DIV bits determine the external clock (ECP clock2) frequency as a ratio of VBUS clock (VCLK) as ECP2CLK = $\frac{\text{OSCIN or VCLK}}{(\text{ECPDIV} + 1)}$ **Note: The device does not guarantee ECPCLK2 beyond 20 MHz for (EMI) purposes.**

Embedded SRAM (eSRAM)

This chapter describes the behavior and specification of the eSRAM wrapper for the TMS470M Series of microcontrollers.

Topic **Page**

5.1 General Description 180
5.2 Block diagram 181
5.3 Module operation. 184
5.4 Bit Access Operation 185
5.5 Memory Fault Detection 186
5.6 Hardware RAM Initialization 192
5.7 Control Registers 193

5.1 General Description

The eSRAM wrapper is used to connect the AHB bus to:

- ACE BRD RAM product
- ACE MVTE RAM product

The eSRAM wrapper controls the RAM operation (READ,WRITE) as well as decoding operation for the RAM space.

The eSRAM supports memory fault correction/detection via internal SECDED circuit. eSRAM wrapper is configurable to include or not include SECDED logic during circuit synthesis.

The eSRAM wrapper supports burst operation as single access.

The eSRAM wrapper uses the 64-bit wide bus of AHB. It allows read and write accesses in 64-bit, 32-bit, 16-bit or 8-bit. However, a parameter within the VHDL file allows the generation of a 32-bit AHB interface.

The eSRAM wrapper supports from 8 KB up to 256 KB RAM.

When used with BRD RAM, the eSRAM wrapper uses the page mode to read the memory.

The memory step for the eSRAM wrapper is either 8KB or 16KB granularity. For 8KB granularity, the maximum memory size which can be supported by eSRAM wrapper is 128KB. For 16KB granularity, the maximum memory size which can be supported by eSRAM wrapper is 256KB.

The eSRAM wrapper supports 2 types of wait-states:

- Address pipelining with introduce a wait-state latency. This feature can be removed or added based on the system timing requirements.
- Extension data phase which allows more time for data propagation

Both types can be used and are programmable through the system module.

The eSRAM wrapper also incorporates the necessary muxing for PMT operation.

The eSRAMW wrapper also traces the ARM Read or Write data to the RAM Trace port or EDP module.

The Wrapper supports ODD or EVEN bus placement. If the wrapper is placed in an ODD or an EVEN Bus, the RAM address decoding is taken care by the wrapper.

The Wrapper can also support mixed banks. The last bank can be an 8KB bank in a 16KB configuration.

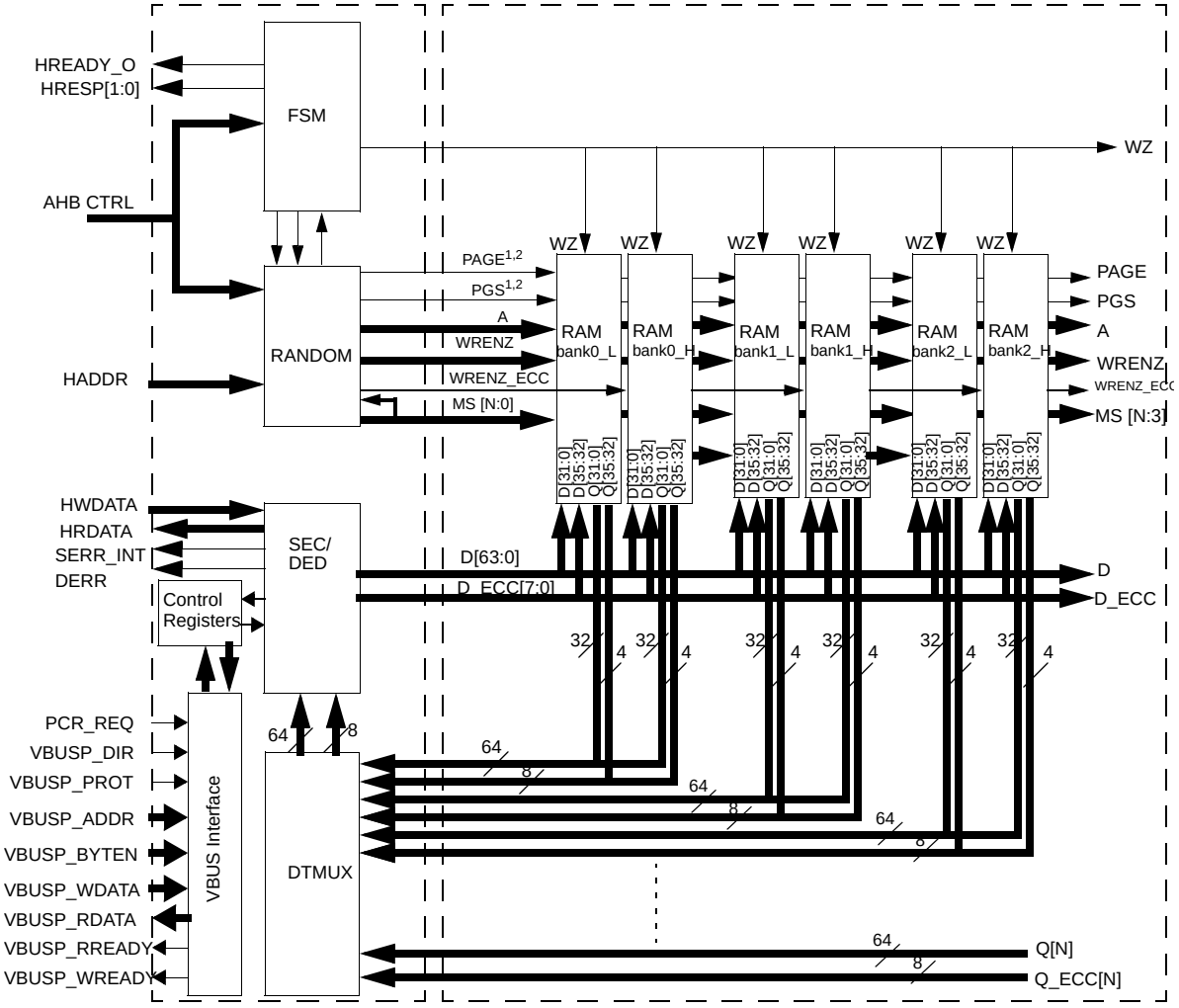
5.2 Block diagram

The eSRAM wrapper is composed of 2 blocks of logic. These 2 blocks are sub-divided as follow:

- Fixed block:
  - FSM module
  - Random logic
  - Datapath mux
- Generic block (re-synthesized per device):
  - RAM arrays
  - SECDED logic
  - Control Registers
  - VBUS Interface

The connections may vary depending of the type of memory used.

Figure 5-1. Block diagram with BRD memories of 2048 word by 36 bit



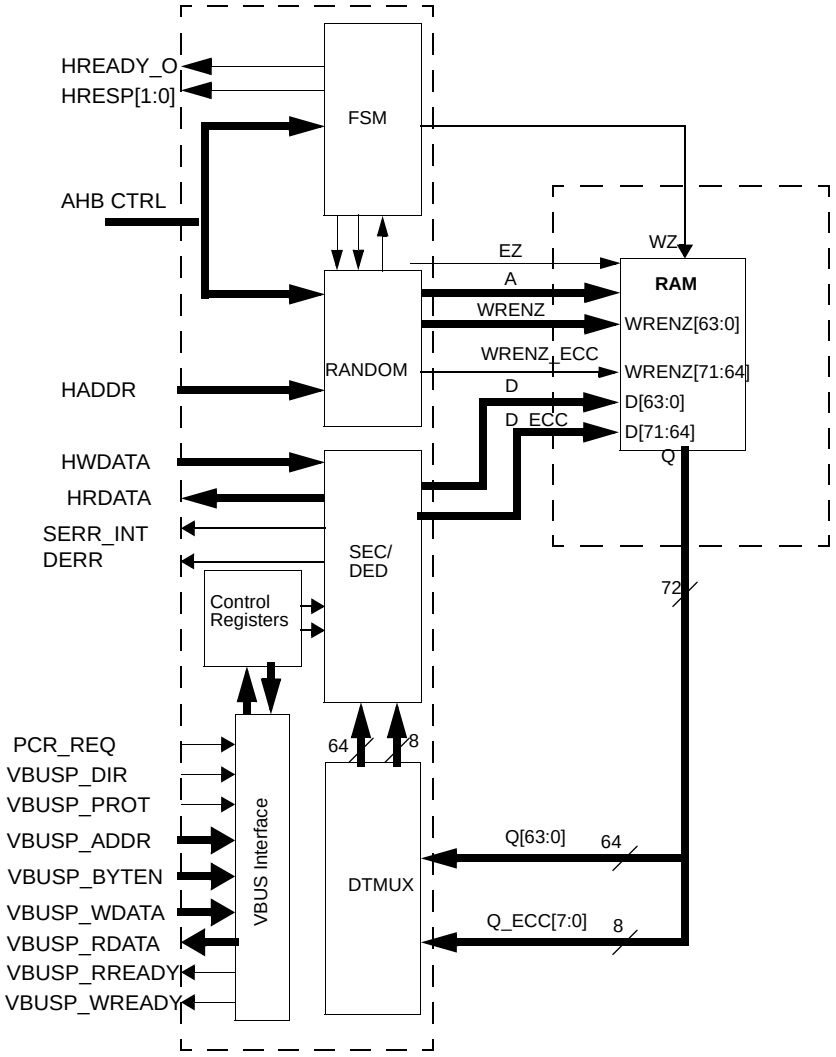
Notes:
1) This memory utilizes a page mode functionality to save power during cycles in which entire word access is not needed. When PAGE is active (PAGE = H), only the lower half or upper half of the data in/out pins (Q and D) are active. When PAGE is active (PAGE=H), and PGS = L, only the lower half of the word is read during a read and written to during a write. If PAGE is active (PAGE=H) and PGS=H, then only the upper half of the word is read from during a read and written to during a write. If PAGE is inactive (PAGE=L), the entire word is read from or written to each cycle (PGS is not used).
2) PAGE and PGS are not used for RAM wrapper configuration with ECC capability. These two signals are tied to zero.

The FSM sub-block is handling the AHB BUS protocol and the RANDOM logic is used to handle address decoding and bus selection. SECDED sub-block generates ECC check bits in both read and write access.

During write access SECDED generates the corresponding ECC check bits for the incoming HWDATA and stores them in the respective ECC memory location. During read access SECDED generates ECC check bits based on the data read from the data memory and compares them with the known good ECC value associated with the read data. SECDED corrects the erroneous data bit in the read data before passing the final data to HRDATA. The DTMUX block receives all the BRD Q outputs from all data and ECC memories and muxes them into SECDED for error correction and detection.

The implementation of RAM arrays at the device level is to use two banks of 2048 x 36 BRD memory per MS[x] select. In this case the D[63:0] from eSRAM wrapper is split into two 32 words with D[63:32] connected to D[31:0] of each bank[x]_H and D[31:0] from eSRAM wrapper connected to D[31:0] of each bank[x]_L. The D_ECC[7:0] is also split into two 4 bit data with D_ECC[7:4] connected to D[35:32] of each bank[x]_H and D_ECC[3:0] connected to D[35:32] of each bank[x]_L. Each bank[x]_L and bank[x]_H share all control signals coming from eSRAM wrapper except D and Q. A total of 32 banks of 2048x36 memory are needed to implement 256k of memory using BRD RAM for 16KB granularity configuration. A total of 32 banks of 1024x36 memory are needed to implement 128k of memory using BRD RAM for 8KB granularity configuration.

Figure 5-2. Block diagram with less than 128Kbyte MVTE memory

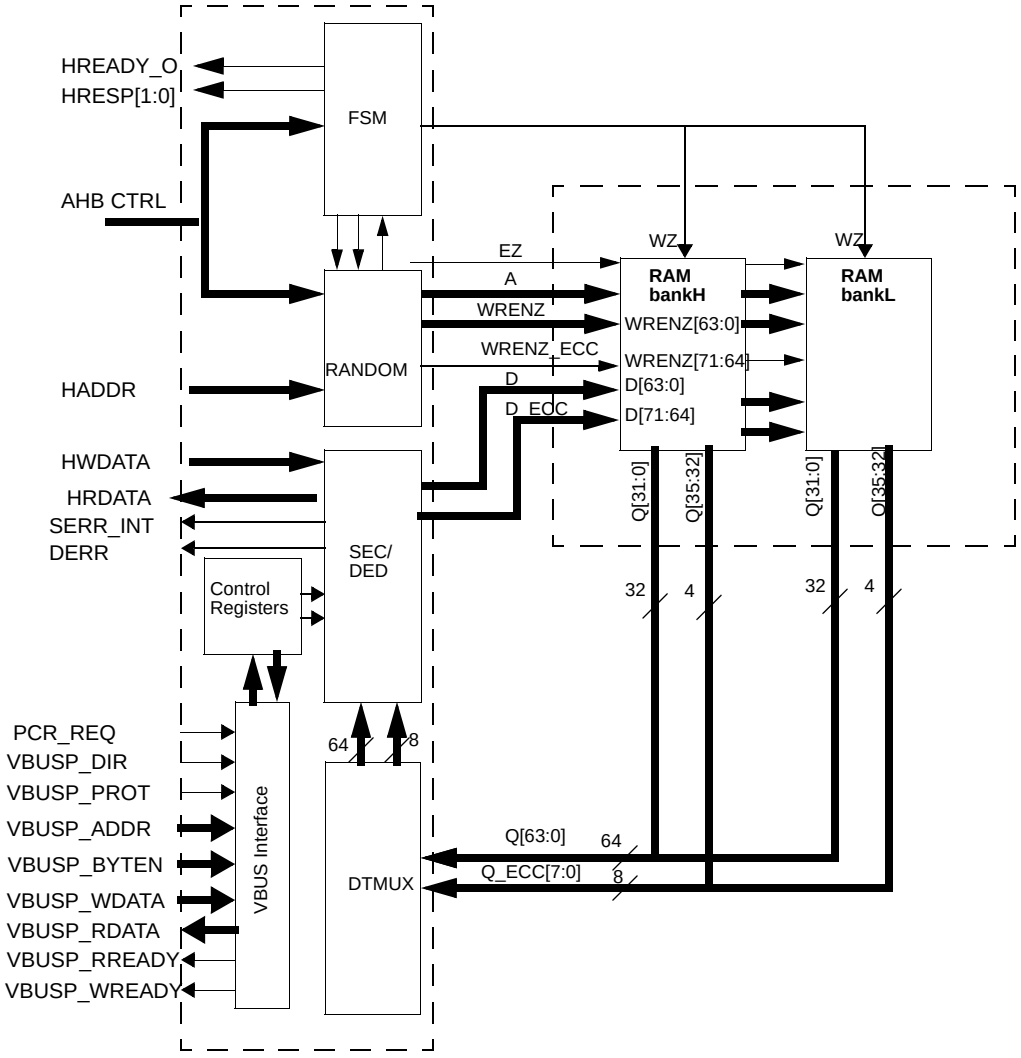


With MVTE memory, the DTMUX block is not needed and the Q bus can be directly connected to the SECDED. However, it is preferable to have it buffered. MVTE memory can be compiled with 72 bit data width

if the total RAM size is less than 128Kbytes and therefore can store both data and ECC in the same memory. For 256Kbytes of memory using MVYE RAM two banks of 32K x 36 bits are needed.

The RAM are instantiated during device integration and include the BIST collar.

Figure 5-3. Block diagram with 256Kbyte MVTE memory



5.3 Module operation

The eSRAM wrapper allows the connection between ACE RAM product with the AHB protocol.

The eSRAM wrapper support access from 8 bit to 64 bit wide in read and write operation. AHB burst are converted into single access transfer type.

The eSRAM wrapper supports zero wait-state access, address pipelined wait-state and extended data phase wait-state during read operation. Write operation are always treated as zero wait-state.

The number of wait-state is programmed via the RAMGCR register in the system module.

The address pipelines wait-state is a removable feature and is supported only if it is present in the wrapper. It is used when address and control signal from the AHB bus are violating the set-up time of the RAM module. The address and control of the bus are latched at the end of the address phase. This feature availability is based on the device requirements. It is not supported on default wrapper configurations. The RAM access is done during the next cycle.

The extended data phase wait-state is used when the timing delay between the RAM and AHB master is long enough to violate the set-up of the master. In this case, the data is output of the RAM is extended by one cycle, which allow the data to propagate to the master within 2 HCLK cycles.

Depending on the operating system clock frequency which is device dependent, the address pipelined wait state and/or extended data phase wait state is enabled. The hardware for Address waitstates support needs to be added for a device based on its timing requirements. Address wait states would not be supported even if it is enabled in RAMGCR but the hardware logic is not present in the wrapper.

Both type of wait-state can be used simultaneously.

When the Module is to be placed on an ODD Address or EVEN Address bus the ODDEVEN BUS generic of the modules is to be used.This helps the Module to Decode the correct ARM address to the RAM.

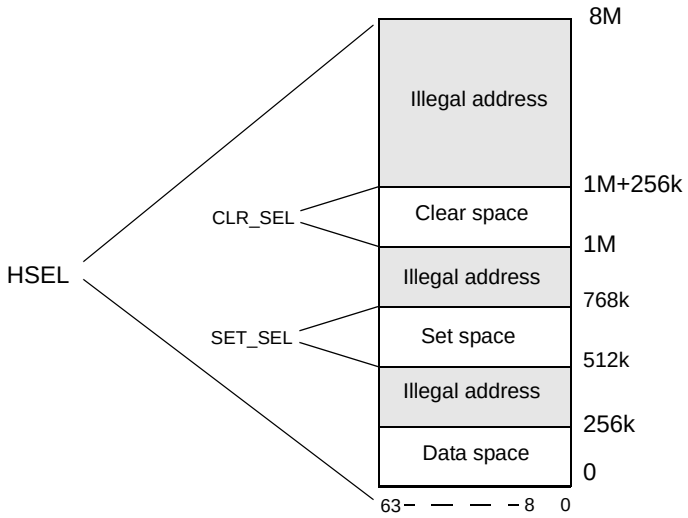
The module is also capable of treating the last bank as only a 8KB bank when the bank size is set to 16KB. This helps in minimizing the NRAMs (number of RAM banks) for a device when the requirement is not a multiple of 16KB.

5.4 Bit Access Operation

Even though the CPU does not support atomic read-modify-write, the eSRAM wrapper provides a mechanism to set or to clear a single bit in the RAM array.

- When reading/writing from the RAM base address, accesses are realized in byte, half-word and word as typical access.
- When accessing the RAM from base address + 0x80000, the write access is realized such that bits set to 1 one the data bus will be set to 1 in the RAM array without disturbing the other bits, the same way a read-modify-write from CPU will do. Bits set to 0 on the data bus will be untouched.
- When accessing the RAM from base address + 0x100000, the write access is realized such that bit set to 1 on the data bus will be clear to 0 in the RAM array without disturbing the other bits, the same way a read-modify-write from CPU will do. Bits set to 0 on the data bus will be untouched.

Figure 5-4. Bit Access Memory Map



For instance, assuming a word at address location 0x08000010, with a value of 0x3459.

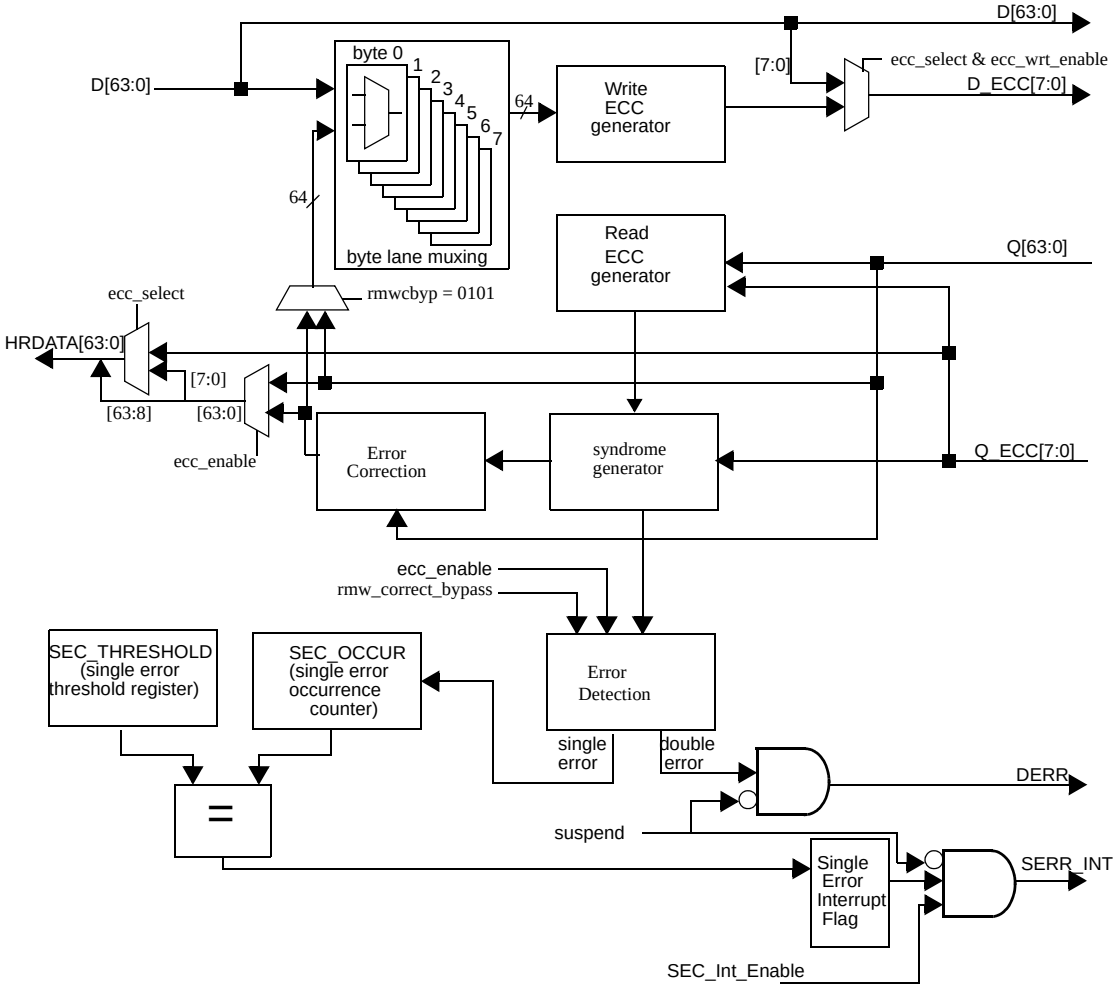
1. Writing the value 0x8315 at address 0x0808 0010 (SET address), will result of the value 0xB75D (0x3459 OR 0x8315).
2. Writing the value 0x8315 at address 0x0810 0010 (CLEAR address), will result of the value 0x3448 (0x3459 AND NOT(0x8315)).

5.5 Memory Fault Detection

eSRAM wrapper offers SECDED (**S**ingle **E**rror **C**orrection and **D**ouble **E**rror **D**etection) capability. After power up reset, the memory detection is disabled through a 4-bit ECC_ENABLE key. After power up reset, the 4-bit key has the reset value of 0x5. Any value but 0x5 will enable SECDED logic. If SECDED is disabled then error correction logic is totally bypassed resulting with best data access time. If SECDED is enabled then data is returned to the host master with slower access time which can affect the data rate at the frequency demanded by the host unless proper wait state is added. Figure 5-5 illustrates the SECDED block diagram.

During memory test mode when PMT_MODE input is high the SECDED is disabled.

Figure 5-5. SECDED Block Diagram



When SECDED is enabled it provides capability to screen out memory faults and correct the fault. SECDED requires a total of eight ECC (**E**rror **C**orrection **C**ode) check bits for each 64 bit of data to be stored in the memory. During write access SECDED automatically generate the ECC bits and store them to the addressed memory. During read access, SECDED automatically generates eight ECC bits based on the 64 bit read data. These eight ECC bits are then compared with the known good ECC value stored in the memory.

5.5.1 Read-Modify-Write Operation

Since the error correcting code is computed on all 64 bits of data, if only a subset of the 64 bits are written, SECDED still requires that all 64 bits are available for the calculation. This means that SECDED will perform a read modify write operation by first reading the entire 64 bits of the addressed location. If there is a single bit error during the read-modify-write operation then the read data is first corrected. The ECC is calculated based on the 64 bit data formed by the un-addressed bytes concatenated with the addressed bytes. After ECC is calculated then write both the 64 bit data and ECC into the memory. An less than 64 bits write access always incur one extra read operation by eSRAM wrapper. With the write buffer the entire read-modify-write

operation is completely hidden from the AHB master as long as there is not another consecutive access. Therefore there is no penalty to AHB host master for write access regardless of access size.

5.5.2 Consecutive Access

When SECDED is enabled and if there is an immediate second access following a less than 64 bits write access then the second access may incur penalty of wait states.

The following table describes the number of wait states incurred to the AHB host master for different types of consecutive accesses.

Table 5-1. Wait state comparison for different access sequences

WST_DENA[a]	WST_AENA[b]	First Access	Wait states incurred to host		Second Access	Wait states incurred to host	
			ECC Disabled	ECC Enabled		ECC Disabled	ECC Enabled
0	0	Read	0	0	Read	0	0
		Read	0	0	Write[c]	0	0
		Write	0	0	Read	1	2[d]
		Write	0	0	Write	0	0
0	1	Read	1	1	Read	1	1
		Read	1	1	Write	0	0
		Write	0	0	Read	1	3[d]
		Write	0	0	Write	0	0
1	0	Read	1	1	Read	1	1
		Read	1	1	Write	0	0
		Write	0	0	Read	2	3[d]
		Write	0	0	Write	0	0
1	1	Read	2	2	Read	2	2
		Read	2	2	Write	0	0
		Write	0	0	Read	2	4[d]
		Write	0	0	Write	0	1[d]

Notes: 1) Extended data phase wait state

2) Address pipelined wait state, if the hardware supports the feature.

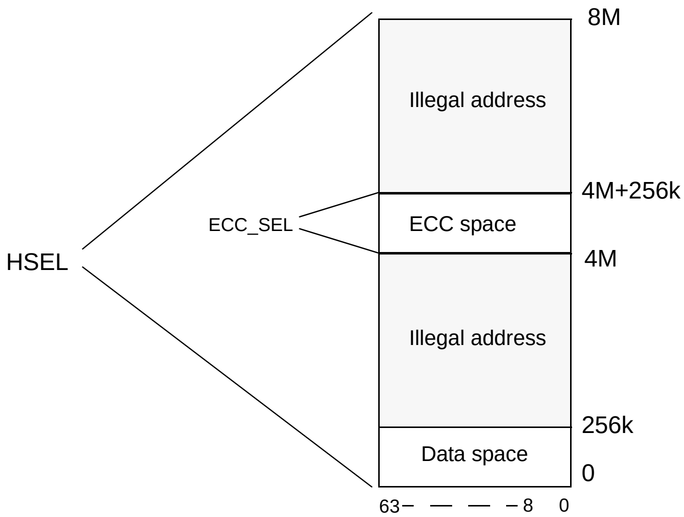
3) All writes in the table are less than 64 bits write access. 64 bit write access does not involve read-modify-write operation and therefore number of wait states pertaining to the second access is identical regardless if SECDED is enabled.

4) Shaded area indicates extra wait state when SECDED is enabled.

5.5.3 ECC Memory Mapping

ECC bits are memory mapped so they can be accessible to the host master for both read and write. Each ECC word is 8 bit wide and resides in a double-word (64 bits) boundary. Each ECC word is offset by 4Mbyte from the corresponding data word to which it is checking. Regardless of SECDED is enabled or not, the ECC bits can always be accessed by the AHB master. No Error Detection or correction takes place when ECC is read.

Figure 5-6. ECC and Data Memory Map



To avoid accidental overriding of ECC bits the write to ECC space is qualified with ECC_WRT_ENABLE bit. If ECC_WRT_ENABLE is not active then writes to ECC space are ignored. During memory test mode when PMT_MODE input is high then write to ECC bits is allowed even if ECC_WRT_ENABLE bit is inactive. ECC bits are always programmed or read using the lower eight bits of either HWDATA[7:0] or HRDATA[7:0] bus.

5.5.4 ECC Generation

The 64 bit data and 8 bit ECC check bits are sent to SECDED for error detection and correction. ECC bits are generated according to modified Hamming code. The calculated ECC bits during a read operation are XORed bitwise with the pre-determined known ECC bits to produce the syndrome. If the data contains no error then the syndrome is zero. Syndrome is decoded to determine whether it is a single bit error or multi-bit error. Syndrome as shown in Table 5-3 also encodes the bit position in error if it is a single bit error.

Table 5-2. *ECC encoding*

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[2]	Checkbits[1]
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	Odd	ECC[7]
x	x	x																	x	x	x	x	x	x	x	x	Odd	ECC[6]
x	x	x									x	x	x	x	x	x	x	x									Odd	ECC[5]
			x	x	x	x	x	x			x	x							x	x	x	x	x	x			Odd	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]

Notes: 1) Each ECC [x] bit represents the parity bit for the corresponding data bits marked with x in the same row.

2) The ECC check bit is generated as either an XOR (Even) or an XNOR (Odd) of the data bits marked with x in the same row.

3) All zeros and all ones data (64 data bits and 8 check bits together) will be detected as error.

Table 5-3. *Syndrome Decode to Bit in Error*

Data Error Bit																																				
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 Error Bit																											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]	Syndrome[1,2,3]
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]	

Notes: 1) Syndrome is a 8 bit value which decodes to the bit in error. The bit in error can be a bit among the 64 data bits or a bit among the 8 ECC check bits.

2) For example, if data bit 30 is in error then the syndrome would indicate as 11110001. If ECC check bit 5 is in error then syndrome would indicate as 00100000

3) Syndrome value of 00000000 indicates there is no error. Any other syndrome combinations not shown in the table are either double bit error or un-correctable multibit error

5.5.5 Double Error Detection

When there are more than one error bit detected the SECDED generates a double error signal on DERR output pin. The address location to which the double error is detected is saved in an error address register.

The raw data from RAM is returned to the CPU. This register is frozen from being updated until it is read by the CPU.

5.5.6 Single Error Correction

SECDED is capable to correct an erroneous bit if it determines the number of error bits is one. Each time a single error is detected and corrected the single error occurrence counter (SEC_OCCUR) is incremented. An single error interrupt is generated when the single error occurrence counter reaches a threshold value. The threshold value is stored in a 16 bit threshold (SEC_THRESHOLD) register. If threshold value is programmed to 1 then both the address and error position of which a single error occurs are captured and saved into registers. The address and error position are frozen from being updated until the single error status flag is cleared by the host system.

An error can be either a hard error or soft error. If the number of correctable single error reaches a large threshold value in a short period of time then it is a high possibility that the error is a hard error. Although a hard single-bit error is correctable, it does increase the risk that additional soft error on the same word can cause a non-correctable error.

5.5.7 False Double Error Detection

When the system comes out of power up reset the content of memory is in unknown state. If SECDED is enabled then either a normal read or the read-modify-write operation forced by the less than 64 bits write access will result in false double error. The read operation will cause ECC to mis-compare since both the data and ECC memory are in unknown state. Several software and hardware practices can be employed to avoid false double error generation.

- Software Memory Initialization
  - After power up reset the system software should first initialize the entire data and ECC memory to known state by resetting all data bits to zero and ECC check bits to 0xFC. If the memory wrapper does not support ECC then data alone is reset to zeroes.
- Enable Read-Modify-Write CORRECTION BYPASS feature
  - When RMWCBYP (Read-Modify-Write correction bypass) is enabled the SECDED bypasses the error detection and correction for the read during the read-modify-write operation forced by the less than 64 bits write access. The bypass feature has no effect on a normal read operation. User can first enable this bypass feature and then writes user data such as constants or interrupt vector table to the memory. ECC bits are generated and written along with the user data to the memory. Upon completion the RMWCBYP should be disabled.

Note: Disable RMWCBYP bits after memory initialization is complete.
If RMWCBYP is not disabled after memory initialization and in the event of a true double error then no double error is generated.

Note: Avoid reading from an un-initialized memory location
If SECDED is enabled then user should avoid reading from any memory location which is un-initialized.

5.5.8 Interrupt and Error Generation

5.5.8.1 Single Error Interrupt

SRAM wrapper can generate interrupt when the number of occurrences of single error is equal to the error threshold value. Interrupt is only generated when ECC_ENABLE != ‘0101’ and RMWCBYP != ‘0101’ and SECINTEN = ‘1’ as shown in Table 5-4

Table 5-4. Single Error Interrupt Generation

ECC_ENABLE	RMWCBYP	SECINTEN	SERR_INT	SECINTFLAG
!= 0101	!=0101	1	Yes	Set
!= 0101	!=0101	0	No	Set
All other combinations			No	Not set

5.5.8.2 Double Error Generation

ESRAM wrapper also set DERR error pin when a double error is detected. DERR is a pulse signal and is sent out of the module.

Note:
When a double bit error is detected, the raw data of the addressed location is passed to CPU. Double bit error signal is sent out of the module.

Table 5-5. Double Error Generation

ECC_ENABLE	RMWCBYP	DERR
!= 0101	!=0101	Yes
All other combinations		No

5.5.9 Emulation

During emulation when SUSPEND signal is high the data read from memory is still passed to SECDED for correction if SECDED is enabled. If a single error is detected then it is corrected but single error interrupt is not generated when threshold is reached. The occurrence counter SEC_OCCUR continues to increment if an correctable error is detected and resets when threshold value is reached.

If a double error address is frozen and is not read by CPU before entering suspend then it remains frozen during suspend even it is read during suspend. If a double error is detected during suspend then the raw data is returned and the error address is not updated and no double error signal is generated.

5.6 Hardware RAM Initialization

eSRAM wrapper has the capability to initialize the entire RAM array to a known state by writing all zeros to the data bits. If ECC is present in the design, the corresponding checkword is also calculated and written to the ECC bits. A pulse signal SYS_MMISTART is asserted by system module to eSRAM wrapper. When the SYS_MMISTART signal is detected, the RAM initialization takes place until the entire memory array is initialized. To speed up initialization time, the wrapper should initialize all memory banks at the same time. When initialization is complete, it should signal to system module with ESRAMW_MMIDONE. This signal is pulse which lasts for at least one VCLK cycle. The bus master would be held in waitstate untill the completion of auto initialisation, but write/read initiated while ESRAMW is doing auto initialisation would not give correct data.

- Write will not be done to the RAM.
- Read may return incorrect data.

5.7 Control Registers

eSRAM control registers are accessed through VBUSP interface. The control register frame is selected when PCR_REQ is high. All registers are 16 bits and reside on 32 bit word boundries. Read and Write access in 8,16 and 32 bit are supported. The base address for the control registers is 0xFFFF F900.

All accesses in suspend mode are converted to privileged mode access. Write always takes one VBUS cycle and read takes two VBUS cycles to VBUS master.

All reserved bits are read as zeros. Write to reserved fields has no effect.

Note:
RAMMSERRADDR, RAMSERRPOSITION, and RAMDERRADDR registers are not reset.

Table 5-6. *eSRAM Control Register*

Offset address Register	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00 RAMCTRL page 194	Reserved			RMWCBYP				ECC_ WRT_ ENA	Reserved				ECC_ENABLE[3:0]			
0x04 RAMTHRESHOLD page 196	SEC_THRESHOLD															
0x08 RAMOCCUR page 197	SEC_OCCUR															
0x0C RAMINTCTRL page 198	Reserved															SECIN TEN
0x10 RAMERRSTATUS page 199																SECIN TFLA G
0x14 RAMMSER-RADDR page 200	Reserved	SERRADDR[14:0]														
0x18 RAMSERRPOSI-TION page 201	Reserved							SERRPOSITION[7:0]								
0x1C RAMDERRADDR page 202	Reserved	DERRADDR[14:0]														
0x20 RAMCTRL2 page 203	Reserved												EDACCMODE[3:0]			

5.7.1 RAM Control Register (RAMCTRL)

Figure 5-7. RAM Control Register (RAMCTRL)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			RMWCBYP				ECC_WRT_ENA	Reserved				ECC_ENABLE[3:0]			
-0			RWP-0x0A				RWP-0	-0				RWP-0x05			

U = Undefined; R=Read, WP=Write in privilege mode,-*n* = Value after reset

.

Table 5-7. RAM Control Register (RAMCTRL) Field Descriptions

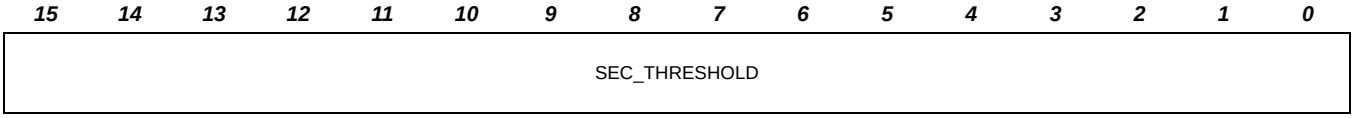
Bit	Name	Value	Description
15–13	Reserved		Reads return zeros and writes have no effect.
12-9	RMWCBYP		Read-modify-write correction bypass.
		0101	ECC correction bypass is enabled.
		all other values	ECC correction bypass is disabled.
			During read-modify-write operation forced by a less than 64 bits write access the read operation goes through error detection and correction as if a normal read when RMWCBYP is disabled. When RMWCBYP is enabled the read operation does not go through error correction and detection. When the device comes out of a power up reset the data and ECC memory are in unknown state. A less than 64 bits write access to the memory will force a read-modify-write operation will result in false double error if ECC_ENABLE is enabled To avoid false double error user can first enable RMWCBYP bit. User data can be loaded to the memory with ECC calculated by SECDED logic. RMWCBYP is then disabled upon completion of user data transfer. Note: The reset value of RMWCBYP is 0b1010 which is to disable ECC correction bypass.
8	ECC_WRT_ENA		ECC Write Enable To avoid accidental write to ECC memory bits the ECC_WRT_ENA must first be enabled before the ECC memory bits can be written.
		0	Write to ECC memory bits is disabled.
		1	Write to ECC memory bits is enabled.
7–4	Reserved		Reads return zeros and writes have no effect.

Table 5-7. RAM Control Register (RAMCTRL) Field Descriptions (Continued)

Bit	Name	Value	Description
3–0	ECC_ENABLE[3–0]		ECC Enable When enabled, ECC is automatically generated by SECDED for both read and write access. During write access, ECC is generated and written to the memory at the same time as the data is generated and written. During read access, ECC is generated based on the read data from memory, and compares it with the known good ECC value stored in the memory. A 1-bit error is corrected before the data is returned to the host. A 2-bit error is signaled to the system module via interrupt.
		0101	The ECC is disabled.
		all other values	The ECC is enabled.

5.7.2 Threshold Register (RAMTHRESHOLD)

Figure 5-8. Threshold Register (RAMTHRESHOLD)



U = Undefined; R=Read, WP=Write in privilege mode,-*n* = Value after reset

.

Table 5-8. Threshold Register (RAMTHRESHOLD) Field Descriptions

Bit	Name	Value	Description
15–0	SEC_THRESHOLD	0–FFFFh	Single Error Correction Threshold This register contains the threshold value for the SEC (single error correction) occurrences before the SERR_INT interrupt is generated. If the threshold is set to one then all single error address and error position are captured in the respective registers.

5.7.3 Occurrence Register (RAMOCCUR)

Figure 5-9. Occurrence Register (RAMOCCUR)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC_OCCUR															
RWP-0															

U = Undefined; R=Read, WP=Write in privilege mode,-*n* = Value after reset

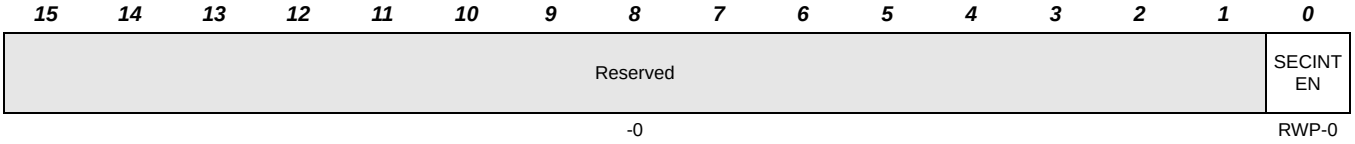
.

Table 5-9. Occurrence Register (RAMOCCUR) Field Descriptions

Bit	Name	Value	Description
15–0	SEC_OCCUR	0–FFFFh	Single Error Occurrence Counter This 16-bit counter contains the number of single bit error occurrences. SEC_OCCUR is reset to 0 when it is equal to the SEC_THRESHOLD value. If the RAMOCCUR value is already higher than the programmed Threshold value then the counter increments till the maximum range and then resets to zero.

5.7.4 Interrupt Control Register (RAMINTCTRL)

Figure 5-10. *Interrupt Control Register (RAMINTCTRL)*



U = Undefined; R=Read, WP=Write in privilege mode,-*n* = Value after reset

.

Table 5-10. *Interrupt Control Register (RAMINTCTRL)* Field Descriptions

Bit	Name	Value	Description
0	SECINTEN		Single Error Correct Interrupt Enable.
		0	Single error interrupt generation is disabled.
		1	Single error interrupt generation is enabled.

5.7.5 Memory Fault Detect Status Register (RAMERRSTATUS)

Figure 5-11. Memory Fault Detect Status Register (RAMERRSTATUS)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															SECINT FLAG
-0															RWP-0

U = Undefined; R=Read, WP=Write in privilege mode,-*n* = Value after reset

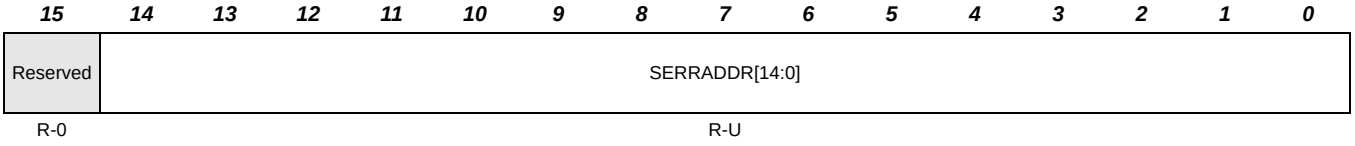
.

Table 5-11. Memory Fault Detect Status Register (RAMERRSTATUS) Field Descriptions

Bit	Name	Value	Description
0	SECINTFLAG		Single Error Correct Interrupt Flag This flag is set when the number of correctable single error reaches the threshold value, even if SECINTEN is disabled. The host clears the flag by writing a 1 to it. If the CPU has not cleared the status bit and another correctable error is detected, then the RAMOCCUR counter will still be incremented.
		0	The number of correctable single errors has not reached the threshold value.
		1	The number of correctable single errors has reached the threshold value.

5.7.6 Single Error Address Register (RAMSERRADD)

Figure 5-12. Single Error Address Register (RAMSERRADD)



U = Undefined; R=Read, WP=Write in privilege mode,-*n* = Value after reset

Table 5-12. Single Error Address Register (RAMSERRADD) Field Descriptions

Bit	Name	Value	Description
15	Reserved		Reads return zeros and writes have no effect.
14–0	SERRADDR[14–0]	0–7FFFh	Single Error Address SERRADDR records the double word address at which an memory fault error is detected. SERRADDR captures the error address when detecting a single bit error if the error threshold value stored in SEC_TRESHOLD register is equal to 1. SERRADDR is frozen from being updated until SECINTFLAG is cleared by the CPU. SERRADDR contains the double word address, which means only HADDR[17–3] are saved. To convert to the byte address, simply left-shift SERRADDR by 3 bits. Note: This register is only reset during power-up reset.

5.7.7 RAM Error Position Register (RAMERRPOSITION)

Figure 5-13. *RAM Error Position Register* (*RAMERRPOSITION*)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							ERRTYPE	SBZ		SERRPOSITION[5:0]					
R-0							R-0	R-0		R-0					

U = Undefined; R=Read, WP=Write in privilege mode,-*n* = Value after reset

Table 5-13. *RAM Error Position Register* (*RAMERRPOSITION*) Field Descriptions

Bit	Name	Value	Description
15-9	Reserved		Reads return zeros and writes have no effect.
8	ERRTYPE		Error Type This bit indicates whether the single error detected is a data bit error or check bit error (ECC bit).
		0	The error is a data bit error.
		1	The error is a check bit error.
7–6	SBZ	0	These bits always read 0.
5–0	SERRPOSITION[5–0]	0–3Fh	Single Error Position SERRPOSITION records the binary encoded error position at which a single error is detected. The error position is captured into SERRPOSITION when a single bit error is detected and the error threshold value stored in SEC_TRESHOLD register is equal to 1.

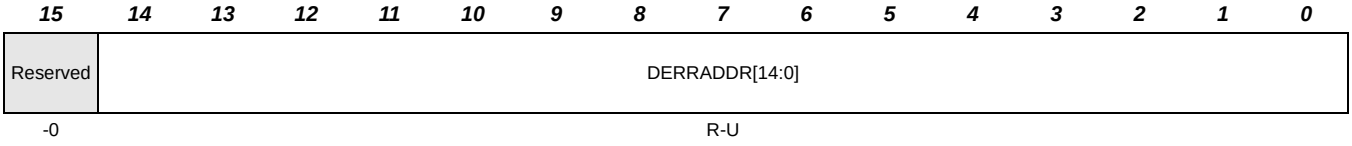
The following examples illustrate the error position being captured in RAMERRPOSITION.

0_0000_0000	Data bit[0] is in error.
0_0001_1111	Data bit[31] is in error.
0_0011_1111	Data bit[63] is in error.
1_0000_0000	ECC bit[0] is in error.
1_0000_0111	ECC bit[7] is in error.

Note:
This register is only reset during power-up reset.

5.7.8 Double Error Address Register (RAMDERRADD)

Figure 5-14. Double Error Address Register (RAMDERRADD)



U = Undefined; R=Read, WP=Write in privilege mode,-*n* = Value after reset

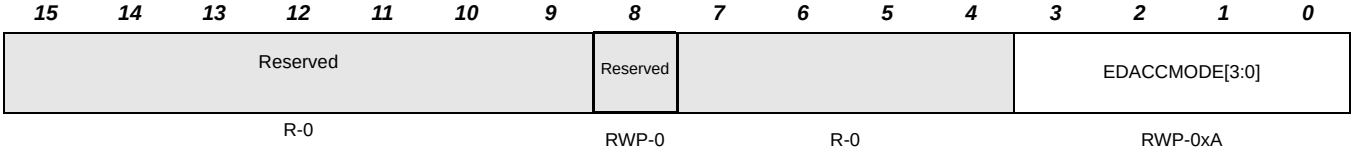
Table 5-14. Double Error Address Register (RAMDERRADD) Field Descriptions

Bit	Name	Value	Description
15	Reserved		Reads return zeros and writes have no effect.
14–0	DERRADDR[14–0]	0–7FFFh	Double Error Address DERRADDR records the double word address at which a memory fault error is detected. DERRADDR captures the error address when detecting a double bit error. DERRADDR is frozen from being updated until DERRADDR is read by the CPU. DERRADDR contains the double word address, which means only HADDR[17–3] are saved. To convert to the byte address, simply left shift DERRADDR by 3 bits.

Note:
This register is only reset during power-up reset.

5.7.9 RAM Control Register (RAMCTRL2)

Figure 5-15. *RAM Control Register* (*RAMCTRL2*))



U = Undefined; R=Read, WP=Write in privilege mode,-*n* = Value after reset

Table 5-15. *RAM Control Register* (*RAMCTRL2*) Field Descriptions

Bit	Name	Value	Description
15-9	Reserved		Reads return zeros and writes have no effect.
8	Reserved		Do not write to this bit.
7-4	Reserved		Reads return zeros and writes have no effect.
3-0	EDACCMODE		Error Detection Mode Disables the Single Error Correction logic of SECDED scheme, However Error Detection is enabled and all double error and single error will be detected, if ECC is enabled in RAMCTRL register.On reset Single Error Correction is enabled.
		0101	SECDED Detection alone. In this mode only detection is enabled.
		all other values	SECDED correction and Detection is enabled.

Phase-Locked Loop (PLL) Clock Module

This device contains a single Phase-Locked Loop (PLL) clock module, the Frequency Modulated zero- pin PLL (FMzPLL). The FMzPLL is available as Clock Source 1 and is intended to be used as the device's main clock source. This chapter describes the functionality of the Frequency Modulated zero-pin Phase-Locked Loop (FMzPLL) and the clock monitor module.

Topic **Page**

6.1 Device Clock Overview 206
6.2 FMzPLL Introduction/Feature Overview 208
6.3 FMzPLL Operation 209
6.4 FMzPLL Control Registers 215
6.5 FMzPLL Calculator (F035 FMzPLL Calculator) 220
6.6 FMzPLL Configuration Example 221

6.1 Device Clock Overview

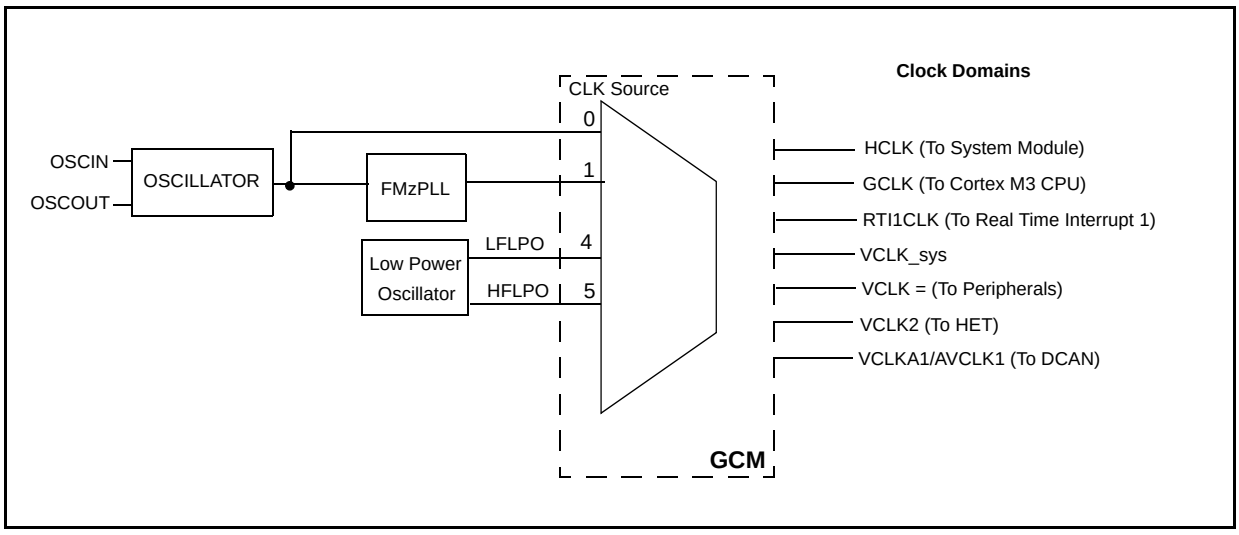
The TMS470M architecture supports a total of seven separate clock sources. This device implements four of the possible seven clock sources including the oscillator, the FMzPLL, the Low Frequency Low Power Oscillator (LFLPO), and the High Frequency Low Power Oscillator (HFLPO). The clock source implementation is shown in the Clock Sources Table below.

Table 6-1. Clock Sources Table

Clock Source #	Clock Source Name
Clock Source 0	External Oscillator
Clock Source1	PLL1 (FMzPLL)
Clock Source 2	Not Implemented
Clock Source 3	Not Implemented
Clock Source 4	Low Frequency LPO (Low Power Oscillator) clock (LF OSC)
Clock Source 5	High Frequency LPO (Low Power Oscillator) clock (HF OSC)
Clock Source 6	Not Implemented
Clock Source 7	Not Implemented

The Global Clock Module (GCM) is used to configure and provide the clocks generated by the PLLs (and other clock sources) to the different modules in the device. A block diagram of the global clock module that is implemented on this device is shown below. The GCM can be configured by using the Primary System Control Registers located between addresses 0xFFFF FF30 and 0xFFFF FF54. See the Control Registers Chapter for more information.

Figure 6-1. Global Clock Module Block Diagram



The Oscillator serves as the main input to clock source 0 and the FMzPLL. It is comprised of the two external device pins OSCIN and OSCOUT. In most applications a crystal or resonator (between 5MHz and 20MHz) is placed between these two pins. The oscillator is provides the necessary feedback to the external crystal or resonator for oscillation and also converts it's sinusoidal input wave into a square wave before it is supplied to the rest of the device.

Note: Vendor Validation of Crystals/Resonators

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 dvice for optimum start-up and operation over temperature and voltage extremes. The vendor also factors in margins for variations in the microcontroller process.

6.2 FMzPLL Introduction/Feature Overview

This section provides an overview of the Frequency Modulated zero-pin Phase-Locked Loop (FMzPLL) module. The FMzPLL is used to multiply the input frequency to a higher (device operation) frequency than can be conveniently achieved with an external crystal or resonator. Additionally, the FMzPLL allows the flexibility to be able to generate many different frequency options from a given crystal or resonator.

Frequency modulation can be superimposed on the FMzPLL ouptut 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 EMI or noise sensitive applications.

6.2.1 Features

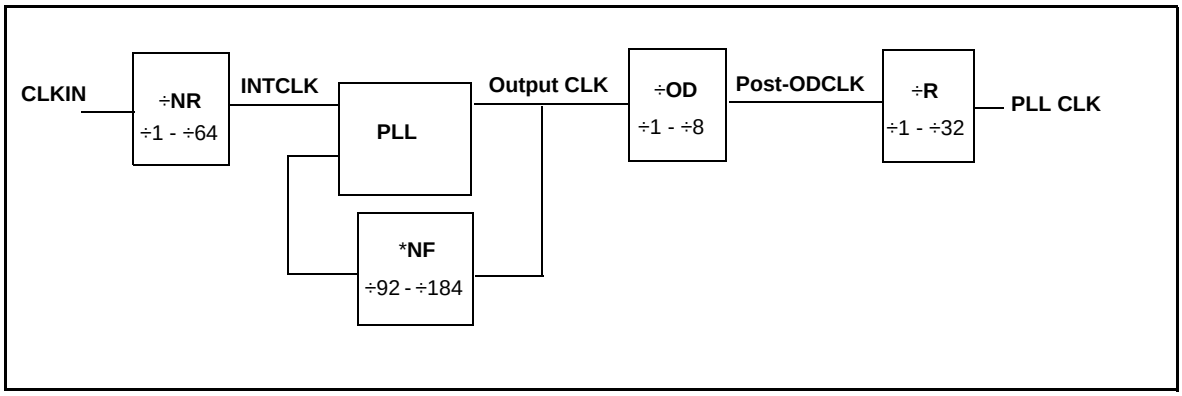
The main features of the FMzPLL clock module are:

- The FMzPLL module can be operated in either modulation or non-modulation mode.
- The FMzPLL prescale allows a wide range of input frequencies for proper operation.
- The phase-frequency detector assures lock to the fundamental reference frequency.
- The FMzPLL provides multiple frequency configuration options.
- When the FMzPLL is used with modulation enabled, the modulation frequency, modulation depth and bandwidth settings are programmable.
- It provides a user-option to reset the device if the external resonator or crystal fails.
- It provides a user-option to either reset the device or bypass the FMzPLL if a PLL Slip is detected.
- A PLL calculator tool (F035 FMzPLL Calculator) is available to assist the user in FMzPLL setup

6.3 FMzPLL Operation

The FMzPLL clock module generates the PLL clock from an external external resonator/crystal reference (CLKIN). The oscillator circuit drives an external crystal/resonator, and the FMzPLL divides (NR), the reference input for a lower frequency input into the PLL (INTCLK); though the input divider has a range from 1 to 64 (integer) INTCLK has a valid range of 1.63MHz to 6.53MHz. The FMzPLL multiplies (NF) this internal frequency by 92 - 184 with a valid range on Output CLK of 120MHz to 500MHz. The FMzPLL output is subsequently divided by two prescale values (OD and R). The value of OD is an integer from 1-8 and R is an integer from 1–32. Optionally, the frequency can be modulated (controlled jitter is introduced). See section 6.3.1.4 for more information about frequency modulation.

Figure 6-2. FMzPLL Block Diagram



Note:
ODPLL must be changed before enabling the FMzPLL

The programming algorithm for the FMzPLL is shown below:

1. Choose the Output CLK frequency as integer divider of output frequency as close to 240MHz as possible. The Output CLK frequency should not exceed 500MHz or fall below 120MHz.
2. Choose the multiplication factor as close to 120 as possible. Multiplier (NF) may not exceed 184 or fall below 92.
3. Select the output divider OD so that the post-ODCLK frequency does not exceed the maximum device frequency.
4. Select the output divider R to further reduce the FMzPLL output frequency provided to the device.

Note: F035 FMzPLL Calculator
The F035 FMzPLL Calculator is also available to assist in configuring the FMzPLL

There will be some delay before changes to the FMzPLL settings will take effect. It is best to disable the FMzPLL prior to changing the settings. All delays shown below are valid once the FMzPLL is enabled. If the FMzPLL is already enabled when the control registers are written, there may be additional delay cycles.

- When NR or NF is changed (or the input clock is enabled), the FMzPLL output is held static for 4096 internal clock cycles ($t = 4096*NR/f_{OSCIN}$)

In general, the smallest delay is achieved when the registers are configured prior to enabling the FMzPLL; when registers are written while the FMzPLL is active, some delays that may be concurrent are treated as sequential. As a specific example, when all parameters are changed once at the beginning of the code, the

FMzPLL output is held static for about 4650*NR/f_{OSCIN}. More information about FMzPLL setup can be found in the FMzPLL Configuration Example.

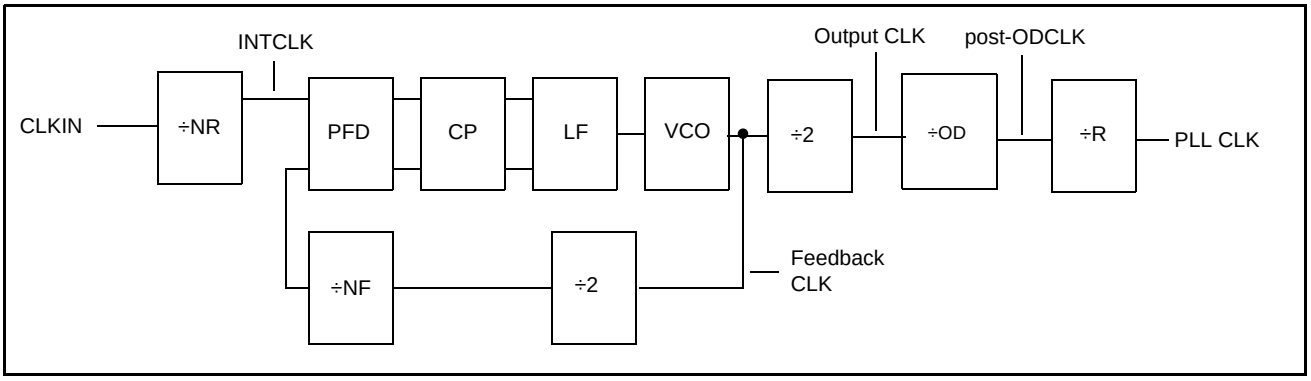
6.3.1 Phase-Locked Loop (PLL) Description

The basic PLL block consists of the following six logical sub-blocks:

- Phase-Frequency Detector (PFD)
- Charge Pump (CP)
- Loop Filter (LF)
- Voltage-Controlled Oscillator (VCO)
- Frequency Modulation
- Slip Detector

Figure 6-3 further illustrates the sub-blocks of the FMzPLL blocks shown in Figure 6-2. 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 greater than the internal frequency (OSCIN/NR). In the forward path (from VCO to PLL CLK), the ÷2 block creates a clean duty cycle.

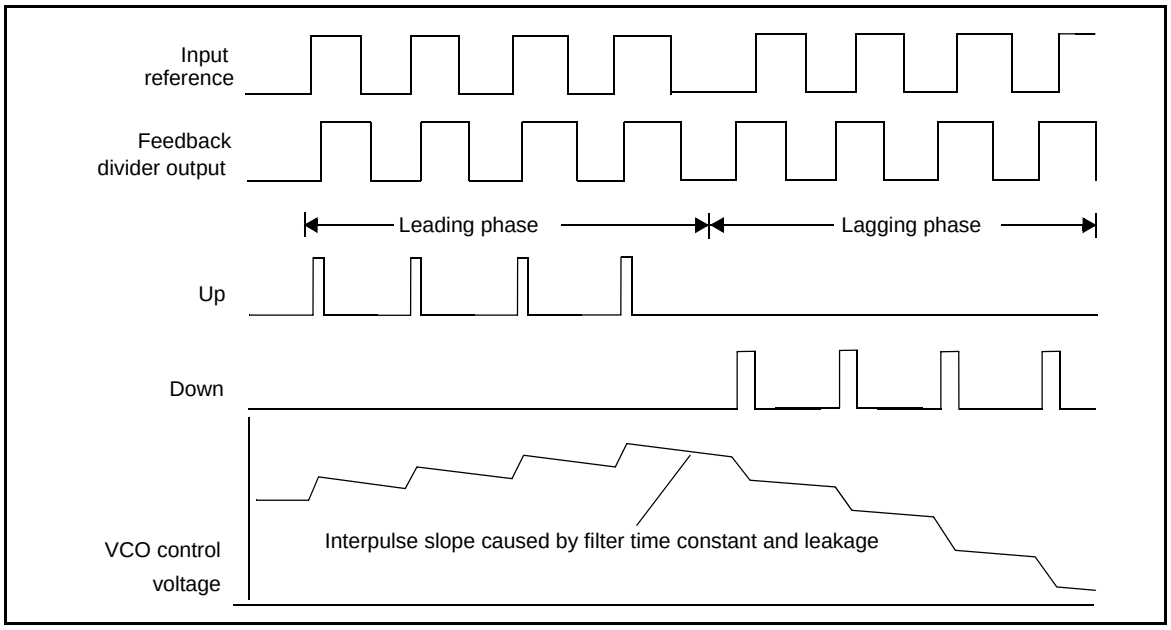
Figure 6-3. Basic PLL Circuit



6.3.1.1 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 6-4.

Figure 6-4. 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 as shown in Figure 6-4. 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 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 bitfield 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 (e.g. 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.

6.3.1.2 Charge Pump and Loop Filter

The charge pump (CP) adds or removes 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.

6.3.1.3 Voltage-Controlled Oscillator

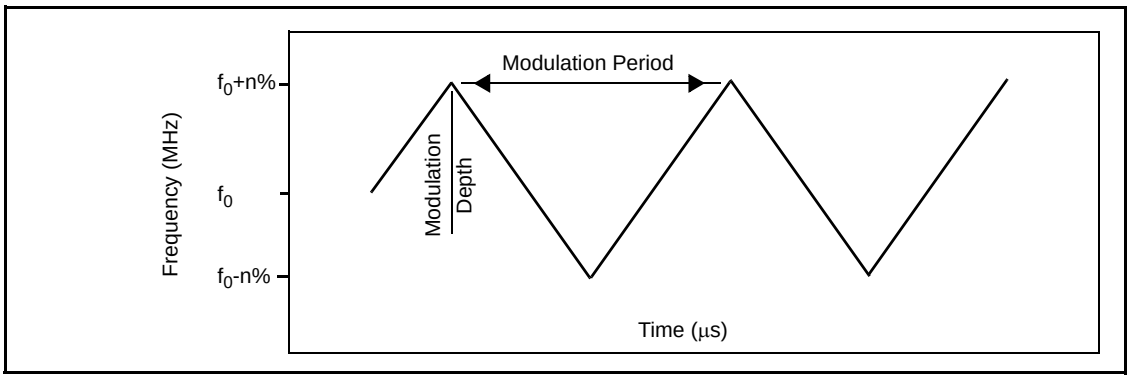
The output frequency of the VCO is proportional to its input control voltage, which is generated by the charge pump via the integrated loop filter. If the VCO oscillates too slowly, the feedback phase begins to lag the reference phase at the PFD, which increases the control voltage at the VCO. Conversely, 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.

6.3.1.4 Frequency Modulation

When the FMzPLL is used in the modulating mode the output clock of the FMzPLL changes frequency in a controlled way, centered around the unmodulated output frequency. The VCO frequency is modulated at the loop filter and creates the triangular frequency modulation as shown in Figure 6-5.

Figure 6-5. Frequency vs. Time



Note: Modulation Frequency and Depth Setting Constraint
There are several combinations of the modulation depth and modulation frequency that are not allowed. Some of these settings effect the FMzPLL even when frequency modulation is not enabled. Refer to the device datasheet to identify these combinations to avoid FMzPLL malfunction.

The programming algorithm for the frequency modulation settings for the FMzPLL is shown below: These settings are controlled by the bits in the PLLCTL2 register.

1. Determine modulation frequency divider NS based on desired spreading (modulation) frequency f_{mod} (f_S).

$$NS = \text{round}\left(\frac{f_{OSCIN}}{NR}\frac{1}{2 \times f_S}\right)$$

$$f_{mod} = f_S = \frac{f_{OSCIN}}{NR}\frac{1}{2 \times NS}$$

2. Determine modulation depth divider (NV) based on desired Modulation Depth.

$$NV = \text{round}\left(5.02154 \times 10^7 \times \frac{\text{Depth} \times f_S}{f_{OutputCLK}}\right)$$

3. Determine the bandwidth divider NB

$$NB = \max\left(\text{ceiling}\left(\frac{0.00406562 \times 10 \times \sqrt{f_{OutputCLK}\frac{f_{OSCIN}}{NR}}}{f_S}\right), 8\right)$$

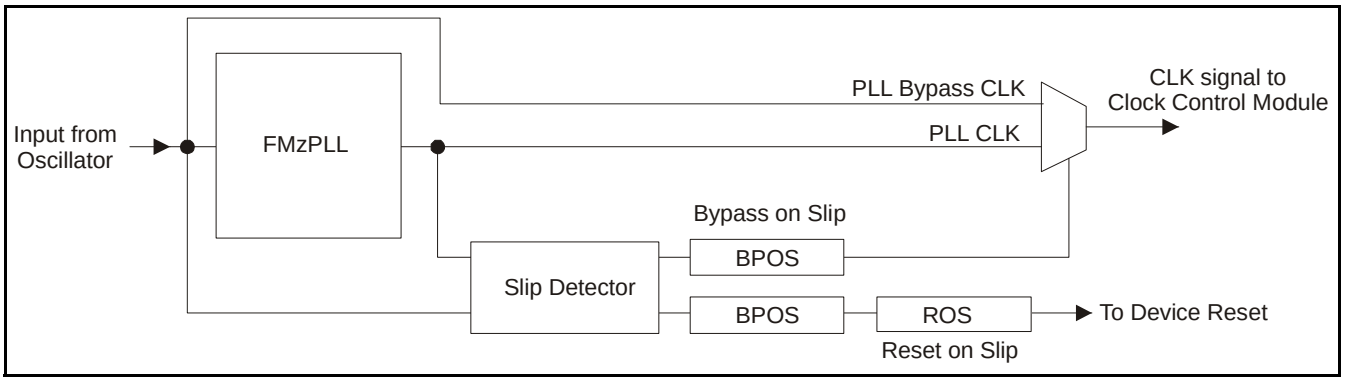
- When NB or NS is changed, the FMzPLL output is held static for 512 internal clock cycles (t = 512*NR/f_{OSCIN}). This timing item cannot be carried out concurrently with a PLL re-lock since the re-lock occurs with NB forced to the default value for faster locking.
- When NV is changed or modulation enabled, the FMzPLL output is held static for 1 - 2 modulation period (t = 1/f_S~1/f_S)

As these different fields (NV, NF, NB, etc.) are changed, the FMzPLL asserts different delays in order to insure a valid output. It is recommended that all pertinent parameters be setup at one time in order to minimize system delays. When multiple parameters are changed simultaneously, it is possible for all delays to be concurrent. As a specific example, when all parameters are changed once at the beginning of the code and modulation is enabled, the FMzPLL output is held static for about 4650*NR/f_{OSCIN} + 1 modulation period. More information about FMzPLL frequency modulation setup can be found in the FMzPLL Configuration Example.

6.3.1.5 FMzPLL Slip Detector

The FMzPLL Slip Detector monitors the input and output of the FMzPLL and reports any 2-cycle slips. Upon a slip detection three actions can be taken: nothing, device reset, or the FMzPLL can be bypassed so that the device is supplied with the oscillator input. The behavior of the device after a PLL slip is detected is configured by the Reset on Slip (ROS) and Bypass on Slip (BPOS) bits in the PLLCTL1 register. The Reset on Slip bit setting determines whether a reset is asserted after a PLL slip is detected. The Bypass on Slip bits determine whether the device will automatically bypass the FMzPLL and use the oscillator to provide the device clock after a PLL slip is detected. See the PLLCTL1 register for more details on the configuration of the ROS and BPOS bits. Figure 6-6 below shows a block diagram of the PLL Slip Detector and the reset/bypass logic.

Figure 6-6. PLL Slip Detection & Reset/Bypass Block Diagram



6.3.1.6 Oscillator Fail Detector

The Clock Monitor Module (CMM) implements circuitry that monitors the oscillator input frequency to ensure that it remains within a specified operational range. (This range is specified in the LPO and Clock Detection section of the device datasheet.) If the frequency of the oscillator ever falls out of the specified frequency window, the clock monitor can either reset the device or switch the device's Clock Source 1 to limp mode. In limp mode all modules driven by Clock Source 1 are switched from the FMzPLL output to be driven by the internal High Frequency LPO (Low Power Oscillator) output. The behavior of the device after an oscillator failure is configured by the Reset on Oscillator Fail (ROF) bit in the PLLCTL1 register. If the ROF bit in the PLLCTL1 register is set when the oscillator fails, a system reset will occur, and the OSC_RST history bit is set in the SYSESR register. The only way OSCFAIL can be cleared (and to re-enable OSCIN as the clock source) is via a power-on-reset.

6.4 FMzPLL Control Registers

6.4.1 Control registers

The FMzPLL clock module has two registers (PLLCTL1 and PLLCTL2) located within the system module, plus it has four bits located in other global control registers of the system module.

The FMzPLL is off at power-on. It may be turned on by clearing the CLK_SR1_OFF bit in the CSDIS register of the System module.

The following sections describe the two registers used to configure the FMzPLL (Clock Source 1). These registers support 8, 16, and 32-bit write accesses.

Figure 6-7. Module 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
0x70 PLLCTL1 Page 216	ROS	BPOS[1:0]		PLLDIV[4:0]					ROF	Reserved	REFCLKDIV[5:0]					
	PLLMUL[15:0]															
0x74 PLLCTL2 Page 218	FMENA	SPREADINGRATE[8:0]								Reserved	BWADJ[8:4]					
	BWADJ[3:0]				ODPLL[2:0]				SPR_AMOUNT[8:0]							

6.4.1.1 PLL Control 1 Register (PLLCTL1)

Figure 6-8 illustrates this register and Table 6-2 provides the bit descriptions.

Figure 6-8. PLL Control 1 Register (PLLCTL1)[Location = 0xFFFF FF70]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ROS	BPOS[1:0]		PLLDIV[4:0]					ROF	Reserved	REFCLKDIV[5:0]					
R/WP-0	R/WP-01		R/WP-1111					R/WP-0	R-0	R/WP-000010					
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PLLMUL[15:0]															
R/WP–0101111100000000															

R = Read, W = Write; P = Privilege Mode, -*n* = Value after reset

Table 6-2. PLL Control 1 Register (PLLCTL1) Field Descriptions

Bit	Name	Value	Description
31	Reset on PLL Slip (ROS)		
		0	Do not reset system when slip is detected
		1	Reset when slip is detected
			Note: BPOS (PLLCTL1.29:30) must be enabled in order to use Reset On Slip (ROS) functionality
30-29	Bypass of PLL Slip(BPOS)		
		10	Bypass on PLL Slip is disabled. If a PLL Slip is detected no action is taken.
		other	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 (PLLCTL1.31) is set to 1, the device will be reset if a PLL slip is detected and BPOS will not bypass the PLL.
28-24	PLL Output Clock Divider (PLLDIV)		R = PLLDIV + 1 $f_{PLL\ CLK} = f_{post_ODCLK}/R$ This divider is depicted as '÷R' in the FMzPLL Block Diagram.
		0x00	$f_{PLL\ CLK} = f_{post\text{-}ODCLK}/1$
		0x01	$f_{PLL\ CLK} = f_{post\text{-}ODCLK}/2$
		:	continues in sequence
		0x1F	$f_{PLL\ CLK} = f_{post\text{-}ODCLK}/32$
			Note: This divider is outside of the PLL macro and can be changed at any time without requiring a re-lock.

Table 6-2. PLL Control 1 Register (PLLCTL1) Field Descriptions (Continued)

Bit	Name	Value	Description
23	Reset On Oscillator Fail (ROF)		
		0	Do not reset system when oscillator is out of range.
		1	Reset when oscillator is out of range.
22	Reserved		Write 0. Do NOT write '1' to this bit.
21-16	Reference Clock Divider (REFCLKDIV)		NR = REFCLKDIV + 1 $f_{INT\ CLK} = f_{OSCIN}/NR$ This divider is depicted as '÷NR' in the FMzPLL Block Diagram.
		0x00	$f_{INT\ CLK} = f_{OSCIN}/1$
		0x01	$f_{INT\ CLK} = f_{OSCIN}/2$
		:	continues in sequence
		0x3F	$f_{INT\ CLK} = f_{OSCIN}/64$
			Note: This value changes the operating point of the PLL; changing this value while the PLL is active requires a re-lock.
15-0	PLL Multiplication Factor (PLLMUL)		Valid multiplication factors are from 92 to 184. NF = (PLLMUL / 256) + 1 $f_{VCO\ CLK} = f_{INT\ CLK} x\ NF$ This multiplier is depicted as '*NF' in the FMzPLL Block Diagram.
		0x5B00	$f_{VCO\ CLK} = f_{INT\ CLK} x\ 92$
		0x5B80	$f_{VCO\ CLK} = f_{INT\ CLK} x\ 92.5$
		0x5C00	$f_{VCO\ CLK} = f_{INT\ CLK} x\ 93$
		:	continues in sequence
		0xB700	$f_{VCO\ CLK} = f_{INT\ CLK} x\ 184$
			The PLL checks the range of the multiplier. IF PLLMUL/256 > 183 OR PLLMUL/256<91, then PLLMUL/256 = 129
			Note: This value changes the operating point of the PLL; changing this value while the PLL is active requires a re-lock.

6.4.1.2 PLL Control 2 Register (PLLCTL2)

The PLLCTL2 register controls the frequency modulated mode of operation of the FMzPLL. The frequency modulation option is available for applications that have critical EMC considerations. Figure 6-9 illustrates this register and Table 6-3 provides the bit descriptions.

Figure 6-9. PLL Control 2 Register (PLLCTL2)[0xFFFF FF74]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FM ENA	SPREADINGRATE[8:0]									Reserved	BWADJ[8:4]				
R/WP-0	R/WP-111111111									R-0	R/WP-00000				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BWADJ[3:0]				ODPLL			SPR_AMOUNT[8:0]								
R/WP-0111				R/WP-001			R/WP-000000000								

R = Read, W = Write; P = Privilege Mode, -*n* = Value after reset

Table 6-3. PLL Control 2 Register (PLLCTL2) Field Descriptions

Bit	Name	Value	Description
31	Frequency Modulation Enable (FMENA)		
		0	Disable frequency modulation
		1	Enables frequency modulation
			Note: The PLL disables the clock output for 1 modulation period when this value is changed while the PLL is active.
30-22	SPREADINGRATE		NS = SPREADINGRATE + 1 $f_{mod} = f_s = f_{INT\ CLK}/(2*NS)$ NS is the modulation frequency divider that is described in the Frequency Modulation Section. $f_{INT\ CLK}$ is the frequency of 'INTCLK' as depicted in the FMzPLL Block Diagram.
		0x000	$f_{mod} = f_s = f_{INT\ CLK}/(2*1)$
		0x001	$f_{mod} = f_s = f_{INT\ CLK}/(2*2)$
		:	continues in sequence
		0x1FF	$f_{mod} = f_s = f_{INT\ CLK}/(2*512)$
			Note: The PLL disables the clock output for 512*NR oscillator cycles if SPREADINGRATE is changed while FMENA is 1. If FMENA is 0, changing SPREADINGRATE has no impact on the clock availability.
21	Reserved		Read/Write, but value has no effect on PLL operation.

Table 6-3. PLL Control 2 Register (PLLCTL2) Field Descriptions (Continued)

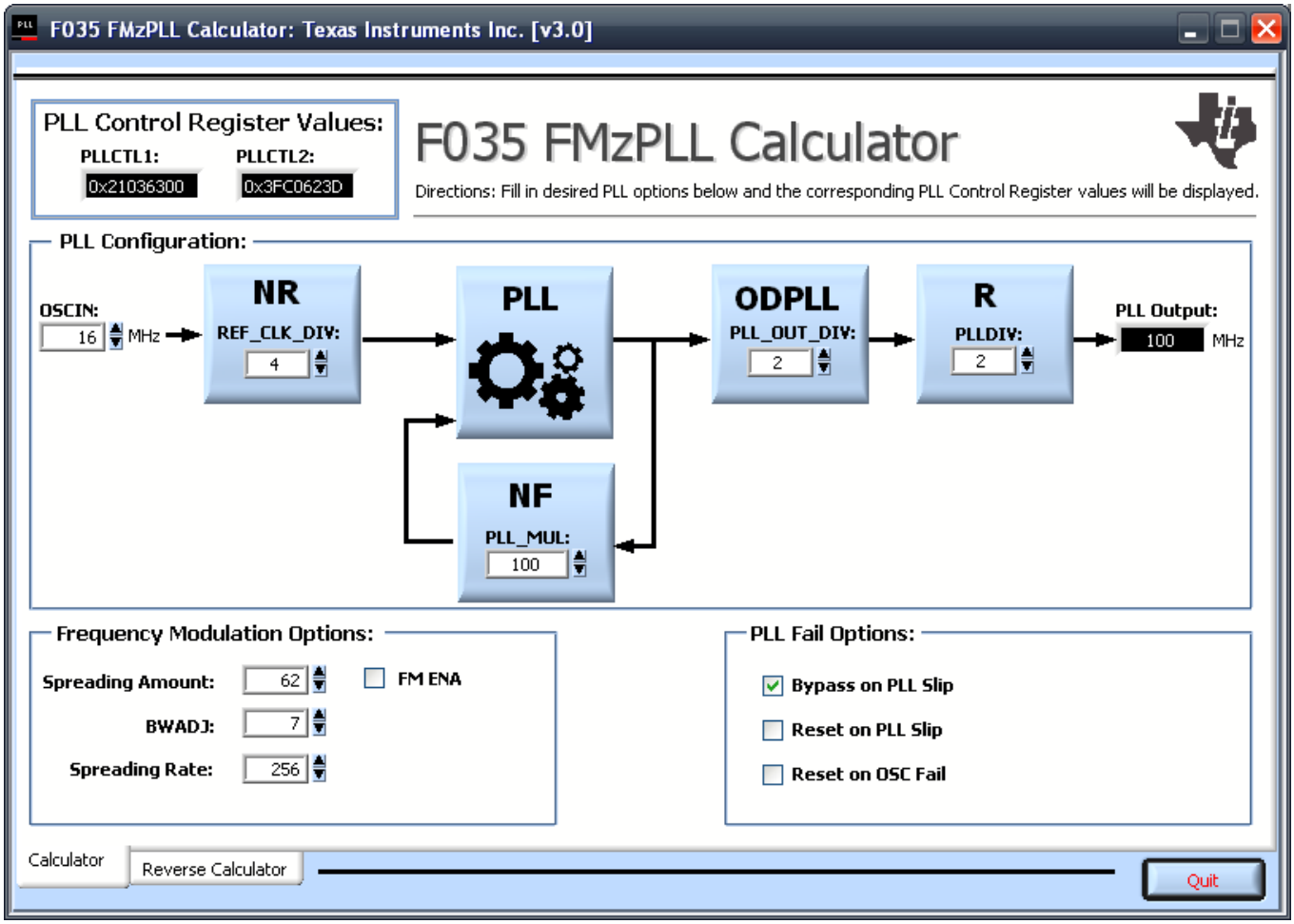
Bit	Name	Value	Description
20-12	Bandwidth Adjustment (BWADJ)		NB = BWADJ + 1 $f_{BW} = f_{nom_BW}/NB$ NB is the bandwidth divider that is described in the Frequency Modulation Section.
		0x007	$f_{BW} = f_{nom_BW}/8$ (must be set to this value in non-modulation mode)
		0x008	$f_{BW} = f_{nom_BW}/9$
		:	continues in sequence
		0x0FF	$f_{BW} = f_{nom_BW}/256$
			The wrapper checks to insure that the BWADJ is programmed within a valid range. IF BWADJ < 7 OR BWADJ > 255, then BWADJ = 7. **Note: The PLL disables the clock output for 512*NR oscillator cycles if BWADJ is changed while the PLL is active.**
11-9	Internal PLL Output Divider (ODPLL)		OD = ODPLL + 1 $f_{post\text{-}ODCLK} = f_{VCO\ CLK}/OD$ These bits must be changed before the PLL is enabled. This divider is depicted as '÷OD' in the FMzPLL Block Diagram.
		0x0	$f_{post\text{-}ODCLK} = f_{VCO\ CLK}/1$
		0x1	$f_{post\text{-}ODCLK} = f_{VCO\ CLK}/2$
		:	continues in sequence
		0x7	$f_{post\text{-}ODCLK} = f_{VCO\ CLK}/8$ **Note: ODPLL cannot be changed while the PLL is active**
8-0	Spreading Amount (SPR_AMOUNT)		NV = SPR_AMOUNT + 1 NV is the modulation depth divider that is described in the Frequency Modulation Section.
		0x000	NV = 1
		0x001	NV = 2
		:	continues in sequence
		0x1FF	NV = 512
			Note: The PLL disables the clock output for 1 modulation period if SPR_AMOUNT is changed while FMENA is 1. If FMENA is 0, changing SPR_AMOUNT has no impact on the clock availability.

6.5 FMzPLL Calculator (F035 FMzPLL Calculator)

The F035 FMzPLL calculator is provided to assist with setting up the PLL. This calculator can be installed and run on a Windows XP based PC. It provides the user with control of the OSCIN speed, multiplier setting, divider settings, frequency modulation settings and also allows the user to set PLL/OSC fail options. Once the user has configured the desired options, the calculator displays the PLL output speed and the corresponding PLLCTL1 and PLLCTL2 register settings. These register settings can then be used to setup the PLL. The tool also contains a reverse calculator mode that will display all PLL options when values for PLLCTL1 and PLLCTL2 are entered. The F035 FMzPLL Calculator is shown below in Figure 6-10.

Note: F035 FMzPLL Calculator Warning
The FMzPLL calculator does not check for all FMzPLL configuration errors. It is the programmer's responsibility to determine if the FMzPLL is configured properly for device operation. Refer to the device datasheet for more information on how to avoid improper device configuration and FMzPLL malfunction.

Figure 6-10. F035 FMzPLL Calculator



6.6 FMzPLL Configuration Example

This section provides an example of how to program the PLL.

Suppose that using a 7MHz crystal, the application requires a

- 140MHz GCLK (and HCLK) frequency
- 134 kHz spreading frequency
- 1% spreading depth.

Then, using the algorithm from Section 6.3

1. Choose Output CLK frequency as integer divider of output frequency near to 240MHz. Output CLK frequency should not exceed 500MHz or fall below 120MHz.

   The integer values for 140MHz are 140MHz or 280MHz. For this example, select 280MHz since it is closer to 240MHz.

2. Choose the multiplication factor near to 120. Multiplier (NF) may not exceed 184 or fall below 92.

   7MHz/NR *(approximately 120) = 280MHz

   7MHz/3*120 = 280MHz

3. Select the output divider OD so that the post-ODCLK frequency does not exceed the maximum frequency of output divider R (device specific frequency).

   If the R-divider can accept 280MHz, then either OD or R can be set to 2 (and the other divider set to 1). If R-divider cannot operate at 280MHz, then the OD-divider must be set to 2.

4. Choose the modulation frequency divider NS

$$NS = \text{round}\left(\frac{f_{OSCIN}}{NR}\frac{1}{2 \times f_S}\right) = \text{round}\left(\frac{7[\text{MHz}]}{3}\frac{1}{2 \times 134[\text{KHz}]}\right) = \{8, 9\}$$

$$f_S = \frac{f_{OSCIN}}{NR}\frac{1}{2 \times NS} = \frac{7[\text{MHz}]}{3}\frac{1}{2 \times \{8, 9\}} = \{145.8, 129.6\}\ [\text{KHz}]$$

   For this example, select 129KHz (with NS = 9).

5. Choose the modulation depth divider NV

$$NV = \text{round}\left(5.02154 \times 10^7 \times \frac{\text{Depth} \times f_S}{f_{OutputCLK}}\right) = \text{round}\left(5.02154 \times 10^7 \times \frac{0.01 \times 0.1296[\text{MHz}]}{280[\text{MHz}]}\right) = 232$$

6. Choose the bandwidth divider NB

$$NB = \max\{\text{ceiling}\left(\frac{0.00406562 \times 10 \times \sqrt{f_{OutputCLK} \times \frac{f_{OSCIN}}{NR}}}{f_S}\right), 8\}$$

$$NB = \text{ceiling}\left(\frac{0.00406562 \times 10 \times \sqrt{280[\text{MHz}] \times \frac{7[\text{MHz}]}{3}}}{0.1296[\text{MHz}]}\right) = 9$$

7. Setting only these fields (that is, not BPOS, ROF, or ROS) yields

   PLLCTL1 = 0x00027700

   PLLCTL2 = 0x820082E7

   INTCLK is 2.33MHz which falls within the permissible frequency range of 1.63 to 6.53 MHz

NF is centered in the range from 92 - 184 at 120

Output CLK has a frequency of 280MHz, falling within the permissible range of 120MHz to 500MHz and near the target of 240MHz.

OD is selected so that post-ODCLK meets the device specification

NB is selected to be large enough to allow modulation frequency f_s.

F035 Flash Module

This chapter describes the behavior and specification of the F035 Flash Module for the TMS470M Series of microcontrollers. The flash electrically erasable programmable read-only memory module is a type of nonvolatile memory which has fast read access times and is able to be reprogrammed in the field or in the application. This section describes the F035 flash module.

Topic **Page**

7.1 Overview 224
7.2 Default flash configuration. 226
7.3 Memory Map 227
7.4 Operation 228
7.5 Control Registers 241

7.1 Overview

The F035 Flash is primarily used to provide the instruction memory to the Cortex-M3 CPU. It can also be used as static data memory such as calibration tables. The flash can be electrically programed and erased many times to ease code development.

7.1.1 Features

- Supports up to 2 banks and 640 kbyte total.
- Read, program and erase with a single 3.3v supply voltage.
- Supports error detection and correction.
  - Single Error Correction and Double Error Detection (SECDED)
  - ECC (Error Correction Code) is evaluated in the flash wrapper
  - 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 time (refer to the device datasheet for details).

7.1.2 Definition of Terms

Terms used in this document have the following meaning:

- 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.
- 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 which share input/output buffers, data paths, sense amplifiers, and control logic. Flash bank can store both program instructions and data.
- Flash module: Flash banks, charge pump, and flash wrapper.
- Flash wrapper (FWM): Power and mode control logic, data path, wait logic, and write/erase state machines.
- 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.
- Standard read mode: The mode assumed when the pipeline mode is disabled. Physically, 128 bits of data (+ 16 bit ECC) is read at a time. However, only 32 bits of data is used while the other 96 bits of data is 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.

7.1.3 F035 Flash Tools

Texas Instruments provides the following tools for F035 flash:

- *nowECC* tool - to generate the flash ECC from the flash data.
- *nowFlash* tool - to erase/program/verify the flash content through JTAG.

  An erased cell reads 1 and a programmed cell reads 0.
- Code Composer Studio V4.x - the development environment with integrated flash programming capabilities.
- Flash API Library - a set of software peripheral functions to program/erase the Flash module. Please refer to *Flash Application Programming Interface User's Specification* for more information.

7.2 Default flash configuration

At power up, the flash module state exhibits the following properties:

- ECC inside flash wrapper is disabled
- Wait states are set to 1
- Pipeline mode is disabled
- The flash content is protected from modification.
- Power modes are set to *Active* (no power savings)

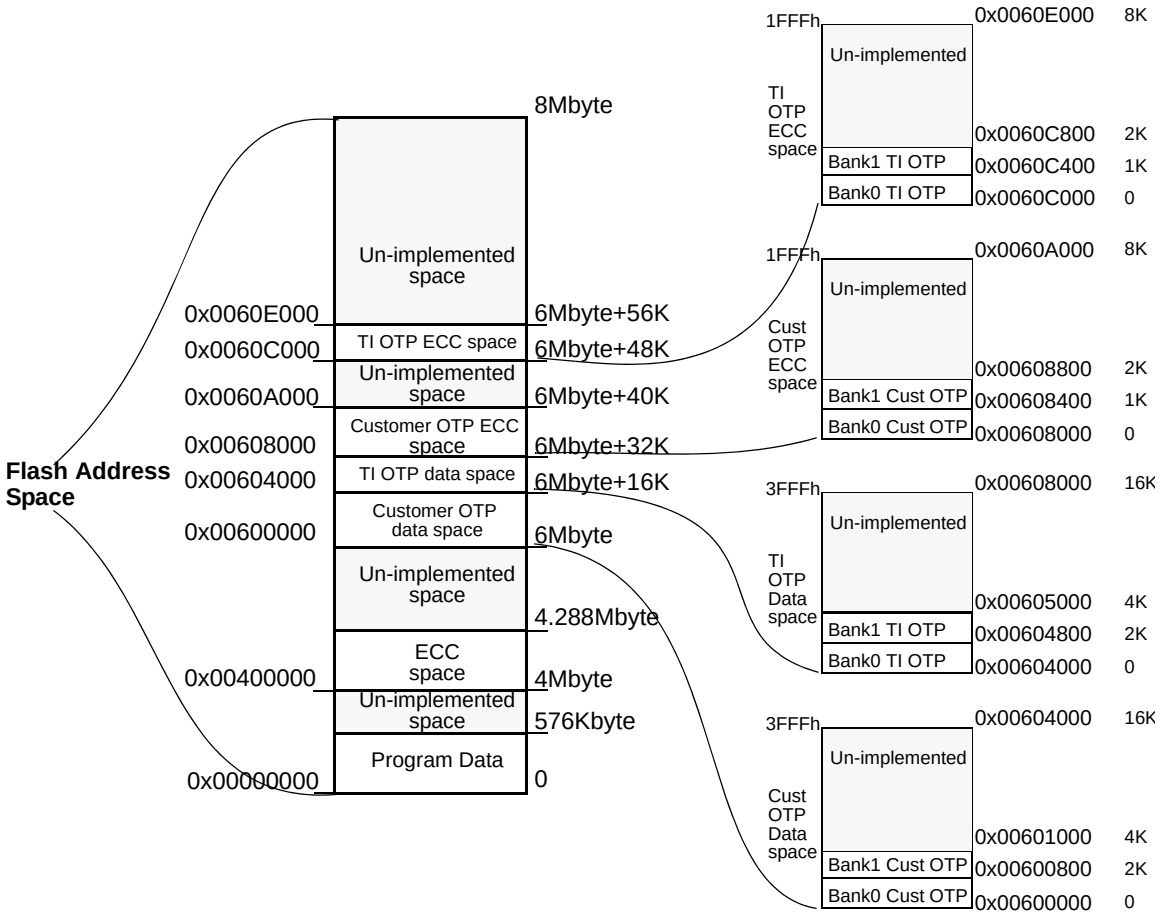
The boot code must initialize the wait states (including data wait states and address wait states) and the desired pipeline mode by by initializing the FRDCNTL register to achieve the optimum system performance. This needs to be done before switching to the final device operating frequency. The required data and address waitstates for timing can be found in the device datasheet.

7.3 Memory Map

The flash module contains the program memory, which is usually mapped starting at location zero, and one Customer OTP sector and one TI OTP sector per bank. The Customer OTP sectors may be programmed by the customer, but can not 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 state machine for erase and program operations.

Usually, the flash memory is located from 0x0 to 0x007FFFFF, total 8 Mbytes. Figure 7-1 shows the typical memory map of a F035 device. Please refer to the device specific datasheet for the Bank/Sectoring information.

Figure 7-1. Flash Memory Map of F035 Device



7.3.1 Illegal Address Generation

Any un-implemented flash space not decoded to a flash bank will cause an illegal address signal to be generated. The illegal address signal forces the CPU to take either a data or an instruction abort.

7.4 Operation

The following sections discuss various modes of operation related to reading, power modes, data protection, and wait state generation.

7.4.1 Flash Read Modes

In addition to *standard read mode*, the flash module also has *pipeline mode*, which affects the technique used to fetch the next memory word. Using this mode allows increased clock speeds and CPU throughput.

7.4.1.1 Standard Read Mode

Standard read mode is defined as the mode in effect when pipeline mode is inactive. Standard read mode is the default read mode after reset. During standard read mode, each read access to flash is decoded by the flash wrapper to fetch the data from the addressed location. The data is returned after the RWAIT number of cycles. RWAIT defines the number of random access wait states. After reset the RWAIT has the reset value of one wait state. Address wait states are not used in standard read mode.

No pipeline buffers are used in standard read mode so every access is used immediately and every access creates a unique flash bank access.

Standard read mode is the recommended mode for lower frequency operation of which RWAIT can be set to zero to provide single cycle access operation. The flash wrapper can operate at higher frequency using non-pipeline mode at the expense of adding wait states. At higher frequencies, it is recommended to enable pipeline mode. Please see the device specific datasheet to determine maximum frequency allowed in standard read mode (non-pipeline mode).

Pipeline mode is disabled by clearing the ENPIPE bit in the flash control register FRDCNTL.0.

7.4.1.2 Pipeline Mode

Pipeline mode removes the flash memory access time for sequential addresses from the critical timing path, which increases clock speeds and CPU throughput.

In pipeline mode, both data wait states (RWAIT) and Address Setup Wait State are required for high frequency operation. In pipeline mode there is always at least one data wait state even when they are set to 0. The Address Setup Wait State is only available in pipeline mode and can be enabled by setting the ASWSTEN bit. Please see the device specific datasheet to determine maximum frequency allowed in pipeline mode versus the number of data and address wait states required.

Pipeline mode is enabled by setting the ENPIPE bit in the flash control register FRDCNTL.0.

7.4.1.3 Read Margin Modes

Read margin modes 0 and 1 are used to test cells for marginality. Read Margin 0 is used to test cells for marginality of programmed cells. Read Margin 1 is used to test cells for marginality of erased cells.

Read margin modes should only be entered when executing from RAM. All banks must be powered up before entering a read margin mode. When entering a read margin mode, or changing from one read margin mode to the other, allow 1us delay before the first flash access to allow the flash banks to adjust to the new mode.

Read margin modes 0 or 1 are enabled by setting the RM0 or RM1 bit in the special read control register FSPRD. The recommended procedures to enter, change and leave read margin mode are shown below:

To enter read margin mode.

1. Set the banks to remain powered up by writing 0xFF to bits 0:15 of FBFALLBACK register.
2. Do one dummy read from each bank to turn on the banks.
3. Start execution out of RAM.
4. Turn on read margin 0 or 1 by writing 1 or 2 to FSPRD register.
5. Flush the data buffer by reading two RAM locations separated by at least 32 bytes.
6. Do two dummy reads from each bank.
7. Wait 1 us

Any reads will now be with the selected read margins. The application can now return to flash if desired.

To change read margin mode to the other

1. Start execution out of RAM.
2. Switch read margin mode to the other by writing 2 or 1 to FSPRD register.
3. Flush the data buffer by reading two RAM locations separated by at least 32 bytes.
4. Do two dummy reads from each bank.
5. Wait 1 us

   Any reads will now be with the selected read margins. The application can now return to flash if desired

To leave read margin mode:

1. Start execution out of RAM.
2. Turn off read margin 0 and 1. Write 0 to FSPRD register.
3. Flush the data buffer by reading two RAM locations separated by at least 32 bytes.
4. Do two dummy reads from each bank.
5. Wait 1 us

   Any reads will now be a standard read. The application can now return to flash if desired. The application can also set the FBFALLBACK register to the original values.

This read-margin mode is normally entered during power-up and a full CRC check of the flash contents is performed. In this way a potential read mode failure can be identified prior to data loss or gain causing a bit flip.

Note:
The read-margin modes are intended for diagnostic capabilities and the flash content is guaranteed for the lifetime specified in the datasheet.

7.4.2 Erase/Program Flash

The F035 Flash should be programmed, erased and verified only by using the F035 Flash API library. These functions are written, compiled and validated by Texas Instruments. The flash module contains a Command State Machine (CSM) to perform program, erase, and validate operations. This section only provides a high level description for these operations, please refer to the *Flash Application Programming Interface User's Specification* for more information.

A typical flow to program flash is:



7.4.2.1 Compact

A device may contain depleted (over erased) columns and/or marginally erased bits. The application code should validate the target flash before erasing it. Otherwise, the leakage current caused by the depleted bits might confuse the sense amplifier in other operation. The CSM provides a "Validate sector" command to perform compaction on the target sector as needed. This command is implemented by the following Flash API function:

Flash_Compact_B().

This function validates the flash data and ECC together. For example, if this function is called to validate the flash data of sector 0 in bank1, it validates both this sector and the corresponding ECC area.

7.4.2.2 Erase

The target flash is ready for erasing after it is validated by the compact function. After erasing, the target flash reads as all '0xFFFFFFFF's. This state is called as 'blank'. The erase function must be executed before programming. The user should NOT skip erase on sectors that read as 'blank' because these sectors may require additional erasing or compaction due to marginally erased bits or depleted columns. The CSM provides an "Erase Sector" command and an " Erase Bank" command to erase the target sector or bank. Similar to the compact function, the erase function erases the data and the ECC together. These commands are implemented by the following Flash API functions:

Flash_Erase_Bank_B() ; //erase the target bank (Support disabling preconditioning)

Flash_Erase_B(); //erase the target sector (Support disabling preconditioning)

Flash_Erase_Sector_B(); //erase the target sector (Do not support disabling preconditioning)

where preconditioning means "Programming 0's prior to applying erase pulses". The application must make sure that the target flash is 'blank' before disabling preconditioning. The Flash API provides the following function to determine if the flash bank is 'blank' before disabling preconditioning erasing:

Flash_Blank_B().

This function can also be used to verify the flash has been properly erased.

7.4.2.3 Program

The CSM provides a "Program customer OTP" command and a "Program Data" command to program the customer OTP area and program data area; The CSM also provides a "Program Customer OTP Check Bits" command and a "Program Check Bits" command to program the customer OTP ECC and data ECC. These commands are implemented by the following Flash API functions:

Flash_Prog_B(); //Program flash ECC and data.

OTP_Prog_B(); //Program customer OTP ECC and data.

Different from the compact and erase function, the program function programs the flash data and ECC separately. The user must generate the flash ECC data (e. g. by calling *the nowECC* tool, which is developed by TI to generate ECC data from a compiled outfile) and program the ECC into the target flash ECC area.

7.4.2.4 Verify

After programming, the user must perform verify using Read Margin Mode. The Verify functions are implemented in the Flash API functions Flash_Verify_B() and Flash_PSA_Verify_B().

7.4.3 ECC Protection

The flash module implemented in this device contains an embedded ECC (**E**rror **C**orrection **C**ode) logic. When enabled, the ECC logic provides the capability of SECDED (**S**ingle **E**rror **C**orrection and **D**ouble **E**rror **D**etection). The ECC logic requires a total of 8 ECC bits for each 64 bits of data to be stored in the flash memory.

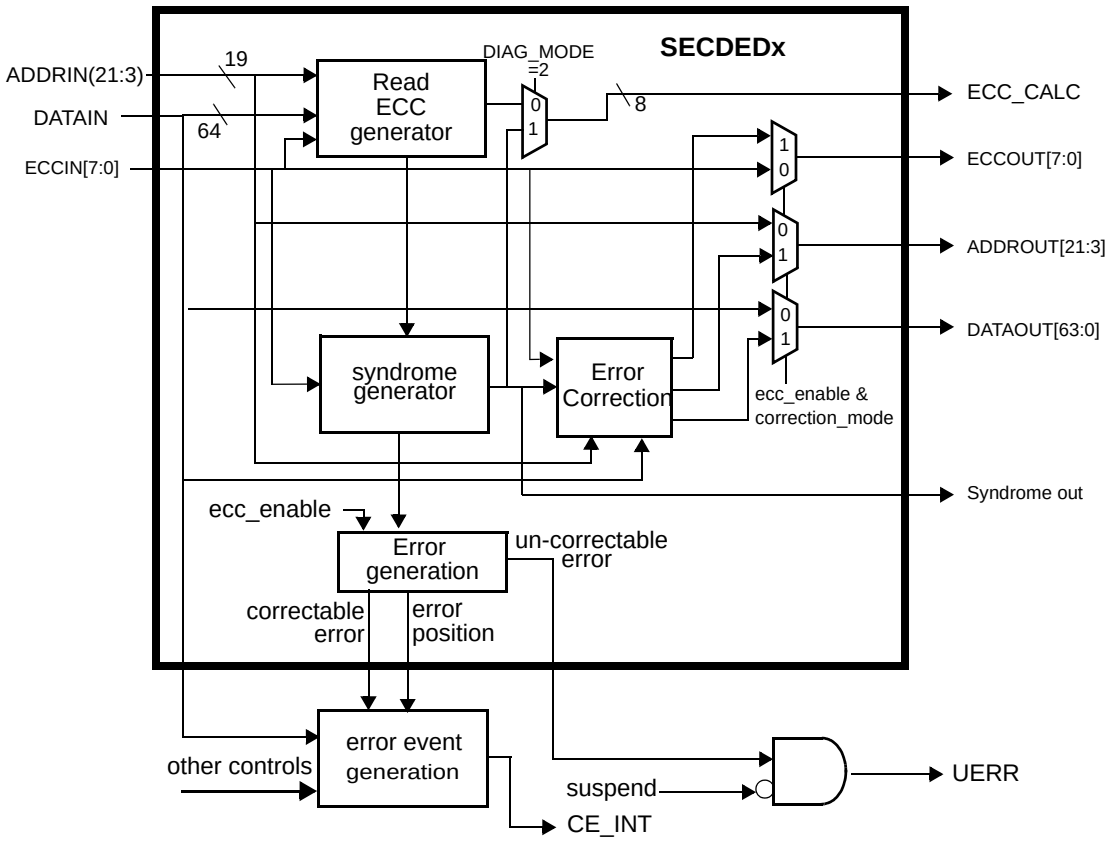
Figure 7-2 illustrates how the ECC logic works. During read operation, the 19 address bits - ADDRIN(21:3), together with the 64-bit data read out of flash banks pass through the 'Read ECC generator' to produce the 8 checkbits. These eight calculated ECC check bits are then XORed with the stored check bits associated with the address and the read data. The 8-bit output is the syndrome. The syndrome is decoded to determine one of three conditions:

a. No error occurred

b. A correctable error occurred

c. A non-correctable error occurred

Figure 7-2. ECC Logic Block Diagram



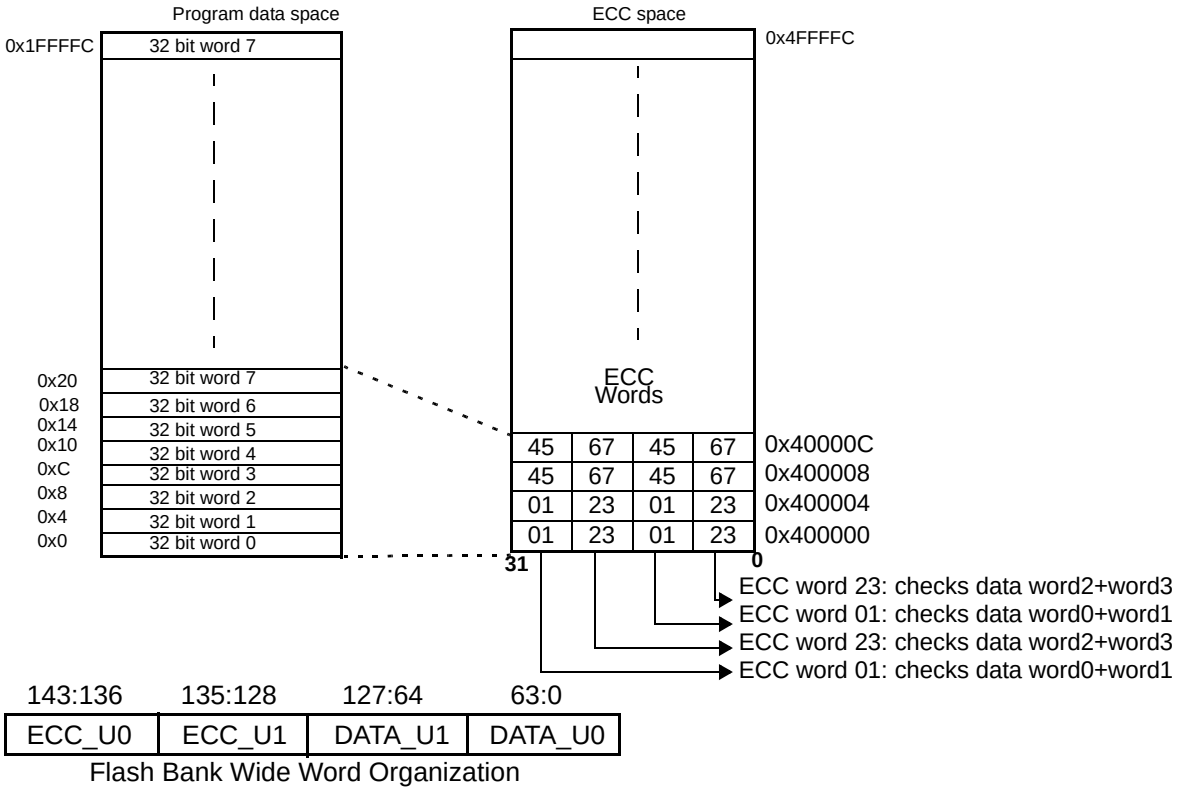
Note:
Since ECC is calculated for an entire 64-bit data, a non 64-bit read such as a byte read or a half-word read will still force the entire 64-bit data to be read and calculated but only the byte or half-word will be actually used by the CPU.

7.4.3.1 ECC Memory Map

In this device ECC bits are mapped to a 4M byte offset from the flash memory base address. Each 64-bit data has its corresponding 8 ECC bits mapped to an address in the ECC space. Refer to Figure 7-3 for an illustration.

As shown in Figure 7-3, the flash bank in this device is 144 bits wide, including 128 bits for normal data width and two sets of 8 ECC bits. Each 8 ECC bits check 64 data bits and 19 address bits. Logically, these 8 ECC bits repeat themselves four times as shown in Figure 7-3. In other words, physically, flash ECC occupies only 1/8 of the corresponding flash data; logically, the size of flash ECC is half of the corresponding flash data.

Figure 7-3. ECC Word Memory Mapping in CPU Address Space



Only 16-bit and 8-bit reads are supported when reading from the ECC space. The flash wrapper cannot support 32-bit reads of the ECC bits because only 16 ECC bits are implemented in one flash row physically. During both 16-bit and 8-bit reads, the flash wrapper will duplicate the physical 16 ECC bits as shown in Figure 7-3.

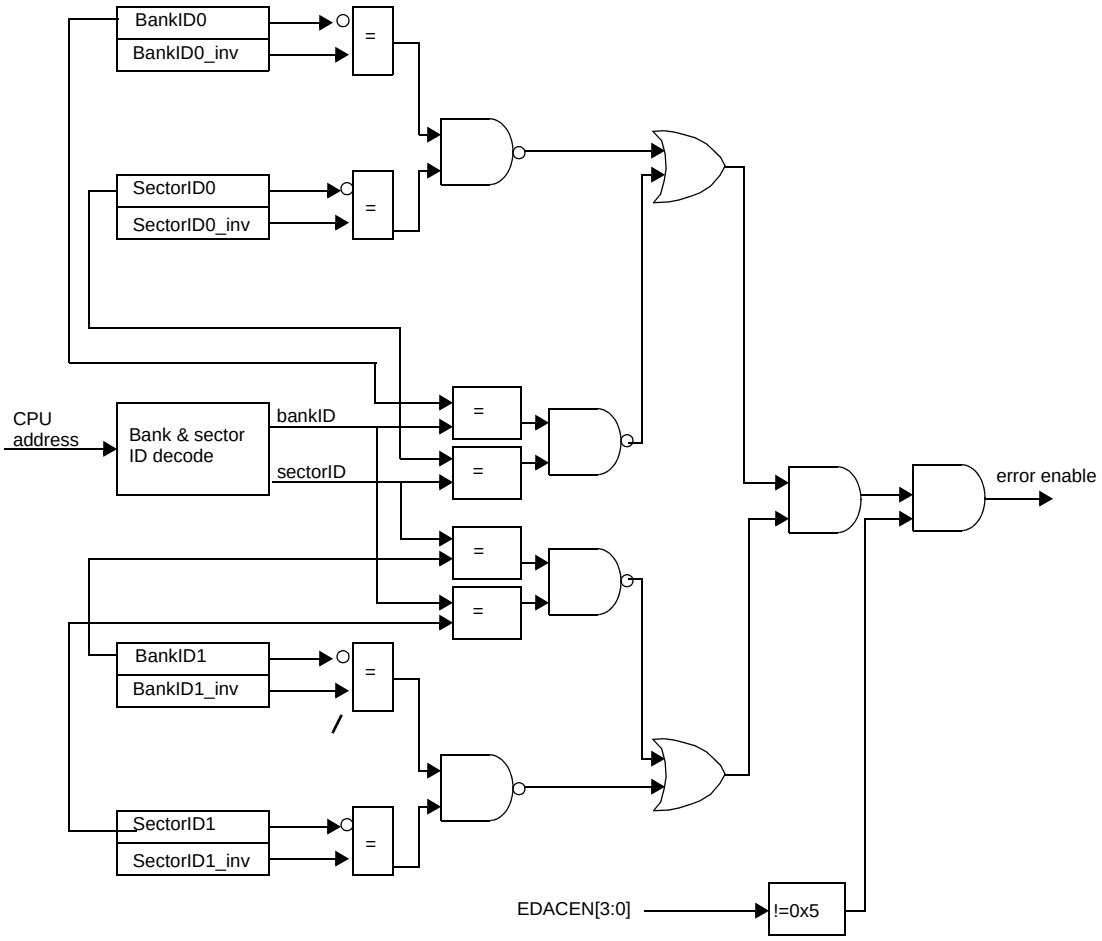
7.4.3.2 Disabling ECC

Globally the ECC logic is enabled and disabled by an 4-bit EDACEN register. When the four bits are 0101, the ECC logic is disabled. Any non-0101 combination value stored in the EDACEN register will enable the ECC logic. It is advisable to enable the ECC logic with 1010 (0xA) stored to the EDACEN to prevent any soft error from disabling ECC logic. ECC logic is enabled by default.

At any given time four different sectors can be disabled from ECC checking. This is done by specifying the bank/sector to be excluded and its inverse value in the sector disable register FEDACSDIS and FEDACSDIS2. Only when the programmed bank/sector ID value and its calculated inverted value matches the programmed inverse value will the sector selected be excluded from ECC checking. After reset the sector register is reset to all zeros which means that no sector is excluded from ECC checking.

.

Figure 7-4. ECC ENABLE Generation



7.4.3.3 ECC Generation Algorithm

The F035 Flash stores 8 ECC bits for every 64 bits of data. This ECC is derived from both address and data values to protect both fields. As shown in Figure 7-5, it takes three steps to generate the 8 ECC bits.

1. Generate the 8 ECC bits for data only
2. Generate the 8 ECC bits for address only
3. XOR the results from 1 and 2 together

Figure 7-5. Flash ECC Calculation

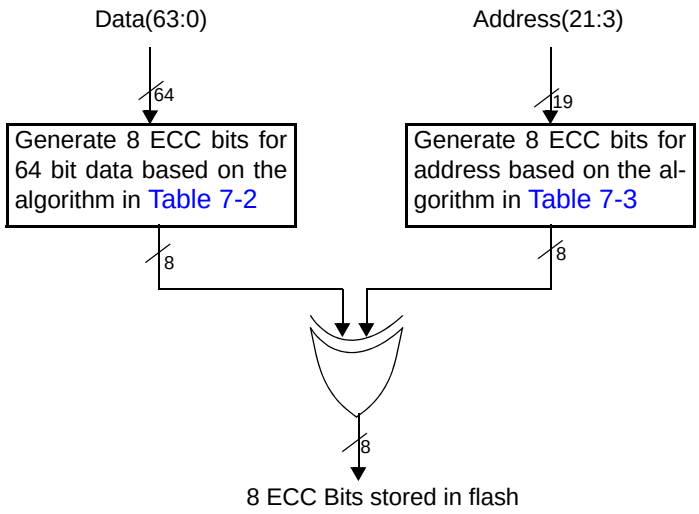


Table 7-1 shows a calculation example:

Table 7-1. Data in Flash (BE32)

ADDR 21:0	Data 31:0	ADDR ECC	Data ECC	Final ECC
0x2415D8	0x01234567	0x97	0xFC	0x6B
0x2415DC	0x89ABCDEF			

The first step is to “Generate the 8 ECC bits for data only”. Table 7-2 shows how to encode the ECC bits for 64 bits of data without address information. Each ECC bit is calculated as the parity bit for the corresponding data bits marked with ‘x’ in the same column. The second row shows the corresponding parity used for each ECC bit calculation. For even parity, XOR all data bits with an ‘x’ across each column to get the corresponding ECC bit. For odd parity, XNOR all data bits with an ‘x’ across each column to get the corresponding ECC bit. For example, ECC bit 7 is xnor_reduce(D55:40,D31:24,D7:0). This could also be calculated as: Sum(D55:40,D31:24,D7:0,1) modulo 2.

For example, the 64 bits of data in Table 7-1 will generate the 8 bit ECC value 0xFC for the data only.

Table 7-2. ECC Encoding inside Cortex-M3 for the Data Bits

ECC bits[1]		ECC[7]	ECC[6]	ECC[5]	ECC[4]	ECC[3]	ECC[2]	ECC[1]	ECC[0]
Parity[2]		Odd	Odd	Odd	Odd	Odd	Odd	Even	Even
Participating Data bits	63		x	x	x		x		x
	62		x	x	x				
	61		x	x		x	x		x
	60		x	x		x		x	x
	59		x	x		x			
	58		x	x			x	x	x
	57		x	x			x		
	56		x	x				x	
	55	x			x	x	x		x
	54	x			x	x		x	x
	53	x			x	x			
	52	x			x		x	x	x
	51	x			x		x		
	50	x			x			x	
	49	x				x		x	
	48	x				x	x	x	x
	47	x		x	x		x		x
	46	x		x	x				
	45	x		x		x	x		x
	44	x		x		x		x	x
	43	x		x		x			
	42	x		x			x	x	x
	41	x		x			x		
	40	x		x				x	
	39		x		x	x	x		x
	38		x		x	x		x	x
	37		x		x	x			
	36		x		x		x	x	x
	35		x		x		x		
	34		x		x			x	
	33		x			x		x	
	32		x			x	x	x	x
	31	x	x	x	x		x		
	30	x	x	x	x				x
	29	x	x	x		x	x		
	28	x	x	x		x		x	
	27	x	x	x		x			x
	26	x	x	x			x	x	
	25	x	x	x			x		x
	24	x	x	x				x	x
	23				x	x	x		
	22				x	x		x	
	21				x	x			x
	20				x		x	x	
	19				x		x		x
	18				x			x	x
	17					x		x	x
	16					x	x	x	
	15			x	x		x		
	14			x	x				x
	13			x		x	x		
	12			x		x		x	
	11			x		x			x
	10			x			x	x	
	9			x			x		x
	8			x				x	x
	7	x	x		x	x	x		
	6	x	x		x	x		x	
	5	x	x		x	x			x
	4	x	x		x		x	x	
	3	x	x		x		x		x
	2	x	x		x			x	x
	1	x	x			x		x	x
	0	x	x			x	x	x	

Notes: 1) Each ECC[x] bit represents the parity bit for the corresponding data bits marked with x in the same column.

2) The ECC bit is generated as either an XOR(Even) or an XNOR(Odd) of the data bits marked with x in the same column.

The second step is to “Generate the 8 ECC bits for address only”. The ECC calculation or algorithm for the address is done as shown in Table 7-3. XOR all address bits with an ‘x’ across each column to get the corresponding ECC(x) bit. For example, ECC bit 7 is xor_reduce(A9,A8,A7,A6,A5,A4,A3). This could also be calculated as: (A9+A8+A7+A6+A5+A4+A3) modulo 2. For example, the address in Table 7-1 will generate the 8 bit ECC value of 0x97 for the address only.

Table 7-3. ECC Encoding for the Address Bits

ECC bits[1]		ECC[7]	ECC[6]	ECC[5]	ECC[4]	ECC[3]	ECC[2]	ECC[1]	ECC[0]
Parity[2]		Even	Even	Even	Even	Even	Even	Even	Even
ADDR	HEX	0007F	7FF80	07F80	19F83	6A78D	2A9B5	0BAD1	554EA
Participating Address bits	21		x			x			x
	20		x			x	x		
	19		x		x				x
	18		x		x	x	x	x	
	17		x	x					x
	16		x	x		x	x	x	
	15		x	x	x			x	x
	14		x	x	x		x	x	
	13		x	x	x	x			x
	12		x	x	x	x		x	
	11		x	x	x	x	x		
	10		x	x	x	x	x	x	x
	9	x						x	x
	8	x					x		x
	7	x					x	x	
	6	x				x			x
	5	x				x	x		
	4	x			x				x
	3	x			x	x	x	x	

Finally, the 8-bit data ECC and the 8-bit address ECC will be XORed together to calculate the final 8-bit ECC value stored in flash. For example, the final 8-bit ECC result for the data / address in Table 7-1 is 0x6B.

7.4.3.4 ECC Syndrome Decoding

As shown in Figure 7-2, during a flash read, when the ECC is enabled, the ‘Read ECC generator’ inside FWM will generated 8-bit ECC value based on the read data and address component. After that, the 8-bit read ECC value will be XORed with the 8-bit ECC generated by the FWM, creating an 8-bit syndrome. The syndrome is decoded to determine if an error has occurred:

- No error

  This is the normal condition. No further action is taken. Note that the error correction logic is always in the datapath and there will be a negative impact on the flash speed regardless of an error occurring or not if ECC logic is enabled.

- Single error correction

  The ECC algorithm is able to correct a single bit error. This erroneous bit could be one of the 64 data bits or one of the 8 ECC bits. The syndrome is decoded and generates a signal to invert the failing bit. In this case, the data read by the CPU is automatically corrected but the data in flash will not be corrected unless the application code fixes the erroneous bit by reprogramming the flash.

  There are three types of interrupts the ECC logic can generate when a single-bit error is detected and corrected:

  1. Interrupt on zero fail

     The most common failure mechanism of flash memory is a data retention loss where electrons drain off of the floating gate. This results in a zero changing into a one. The ECC circuit can detect this type of failure. The failing bit can be reprogrammed without the need to erasing the flash first, thus restoring the electrons to the floating gate. This effectively fixes the flash and reduces the possibility that later on, a non-correctable failure will occur. The address and failing bit of each correctable error is latched and an interrupt can be generated to signal the program that a failure occurred.

When “Error On Zero Fail Only Enable (EZFEN)” bit is set, the “Error On Zero Fail Only” flag can interrupt the processor when an expected 0 value turns into a 1 value. The address and error position are frozen from being updated until the correctable error status flag is cleared by the host system.

Note: Instructions to clear the status flag should be placed at the end of the interrupt subroutine. This is to avoid the address and position registers latching new values during the interrupt subroutine.

2. Interrupt on One Fail

   In the less likely event of data gain, when a failing bit occurs resulting in a one turning into a zero, the flash sector must be erased before reprogramming.

   When the “Error On One Fail Only Enable (EOFEN)” bit is set, the “Error On One Fail Only” flag can interrupt the processor only when an expected 1 value turns into a 0 value. The address and error position are frozen from being updated until the correctable status flag is cleared by the host system

   To generate an interrupt for both zero fail and one fail simply enable both the “Error on Zero Fail Only” and the “Error on One Fail Only” enable bits.

   If both enable bits are disabled then no interrupt is generated but the ECC logic still detects and corrects any correctable error.

3. Interrupt on error profiling

   When the profiling interrupt is enabled by setting EPEN bit, each time a correctable error is detected by the ECC logic, the correctable error occurrence counter (SEC_OCCUR bits in the FCOR_ERR_CNT register) is incremented. A profiling interrupt is generated when the correctable error occurrence counter is equal to the threshold value. The threshold is a 16-bit user programmable value (SEC_THRESHOLD) stored in the FEDACCTRL2 register. The error occurrence counter is frozen in suspend mode (during JTAG debug mode).

   An error can be either a hard error or a soft error. If the number of correctable errors reaches a large threshold value in a short period of time then it is a high possibility that the error is a hard error. Although a hard single-bit error is correctable, it does increase the risk that additional soft errors on the same word can cause a non-correctable failure.

When the ECC logic detects a correctable error, the associated error address and error position are immediately stored into the *Correctable Error Address Register (FCOR_ERR_ADD)* and the *Correctable Error Position Register (FCOR_ERR_POS)*. During pipeline mode, when data are pre-fetched in advanced of a CPU fetch, the interrupt is not generated until the word is requested by the CPU. When the addressed word is requested by the CPU, the associated error interrupt status flag (Error on Zero Fail or Error on One Fail) is set and the interrupt request is generated if the word was detected to have an error.

During pipeline mode the error address and position registers may be overridden with new values if the earlier word detected with an error is not requested by the CPU and another error is detected for the subsequent bank reads. This means that the address and position are overridden as long as the error status flag is not set. Once an error status flag bit is set the error address and position are frozen until the associated error status flag is cleared by the CPU.

If profiling is enabled then any correctable error detected will increment the error counter. The counter is incremented immediately when an error is detected regardless if the faulted word is later accessed or not by the CPU.

- Non-correctable error detection

  When there is more than one error bit detected, the SECDED generates a un-correctable error signal on the UERR output signal to the ESM and ECC_MUL_ERR (Double/Multi Bit ECC error) flag is set in FEDACSTATUS register. During an non-correctable error the data is unchanged. The address location on

which the un-correctable error is detected is saved in an error address register FUNC_ERR_ADD. This register is frozen from being updated until it is read by the CPU. The UERR will set an un-correctable error interrupt flag inside the ESM module. An abort is not generated for a non-correctable error due to timing impact on HREADY generation.

Note: The UERR signal is frozen upon the first occurence of an error. Subsequent errors detected will not assert the UERR until the error address register is read. The freeze can be cleared by reading the FUNC_ERR_ADD register.

When an un-correctable error is detected, the error address is immediately saved to the un-correctable error register. The error signal is not generated until the word is requested by the CPU. Once the error signal is generated the un-correctable error address register is frozen until it is read by the CPU. Similar to the correctable error address register the un-correctable error address register can be overridden with a new value if the previous detected un-correctable data word is not requested by the CPU and another un-correctable error is detected by the ECC logic.

7.4.3.5 JTAG Debug Mode

During JTAG debug mode, the data read from memory is still passed to the ECC logic for correction if ECC is enabled. If a correctable error is detected then it is corrected but the error interrupt is not generated and the error occurrence counter is not incremented if in profiling mode. If an un-correctable error is detected then the raw data is returned without generating an un-correctable error signal.

7.4.4 Data Security

Data security against either accidental or deliberate access by unauthorized agents is built into the flash module. Level 1 security allows each sector to be individually protected from any access other than read.

At power up, all of the sector-enable bits in FBSE are initialized to 0 so that the flash memory location cannot be modified. The sector should only be enabled when erasing or programming flash memory. The sector-enable bits can only be modified in privileged mode **AND** when the PROTL1DIS bit in the FBPROT register is 1. The sector-enable bits of the sectors within the bank selected by bits BANK[2:0] in register FMAC can be modified by writing the corresponding FBSE bit.

7.4.5 Automatic Power-down of Flash Banks

The flash module provides a mechanism to automatically power down flash banks after they have not been accessed for some user programmable time. Special timers automatically sequence the power up and power down of each bank independently of each other. The charge pump module has its own independent power up/down timers as well.

7.4.5.1 Active Grace Period

The active grace period (AGP) can be used to optimize the flash module power consumption vs access time. Faster access times are associated with higher power modes of operation. At one extreme, the power control logic could attempt to reduce power consumption by putting the banks and charge pump into a low-power mode immediately at the end of every flash access. However, if accesses are only a few cycles apart, this can actually increase power consumption versus leaving the flash powered, because the banks and charge pump consume more power during flash startup and access.

The active grace periods (supported for each bank independently in addition to the charge pump module) allow the banks and/or charge pump to be maintained in active mode for a specified period following an access. This is done in anticipation of another read within the AGP time, to allow the subsequent read to have a faster access and spend less time dissipating power than if the bank went into one of the low power modes immediately. If the next access does not occur within the AGP time, the power control logic can automatically put the bank and/or charge pump into a low-power mode to reduce power consumption during long periods of inactivity.

The AGP value is programmed by a set of programmable counters (FBAC and FPAC2) which keep the flash bank or charge pump in active mode until the counter expires, at which time the bank or charge pump reverts to its fallback power mode as defined in the FBFALLBACK and FPAC1 registers. The application software can program the fallback power mode to be standby or sleep mode to reduce power consumption, or program it to be active mode to keep the bank active regardless of counter settings (default). The charge pump AGP counter remains in its initialized state when any one of the banks is active, including the AGP counter of the bank. The charge pump AGP counter begins counting when all banks have become inactive. The application software can also check the current power mode of flash bank and charge pump by reading the FBPRDY register. See register descriptions for detail information.

7.4.5.2 Wakeup of Flash Banks/Pumps

Any access to a flash bank causes the bank and charge pump to go into active mode, regardless of their current state. Also, any erase, program, or validate sector command causes the charge pump to become active.

If the charge pump is in sleep mode when the flash access begins, the power mode control logic automatically sequences the charge pump to standby mode, then to active mode. Also, if any bank is active or in standby mode, the charge pump is active, independent of the charge pump fallback power mode.

The CPU can override the power control functions of the flash module by setting all of the AGP counters to zero. In this case, the power mode control logic still sequences the pump through standby mode automatically if needed, and it activates the pump automatically if any bank is put into any power mode other than sleep mode.

7.5 Control Registers

This section details the flash module registers, summarized in Figure 7-6. 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 0xFFF87000.

Figure 7-6. Flash Control Register Summary

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
0xFFF87000 FRDCNTL Page 245	Reserved															
	Reserved				RWAIT				Reserved			ASW-STEN	Reserved			EN PIPE
0xFFF87004 FSPRD Page 246	Reserved															
	Reserved														RM1	RM0
0xFFF87008 FEDACCTRL1 Page 247	Reserved							SUSP_IGNR	Reserved				EDACMODE			
	Reserved					EOF EN	EZF EN	EPEN	reserved		EOCV	EZCV	EDACEN			
0xFFF8700C FEDACCTRL2 Page 250	Reserved															
	SEC_THRESHOLD															
0xFFF87010 FCOR_ERR_CNT Page 251	Reserved															
	COR_ERR_CNT															
0xFFF87014 FCOR_ERR_ADD Page 252	Reserved							COR_ERR_ADD[26:16]								
	COR_ERR_ADD[15:3]														word offset[2:0]	
0x18 FCOR_ERR_POS Page 253	Reserved															
	Reserved							ECC ERR	SERR_POS[7:0]							

Figure 7-6. Flash Control Register Summary (Continued)

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
0xFFF8701C FEDACSTATUS Page 255	Reserved															
	Reserved							ECC MUL ERR	Reserved					ERR ONE FLG	ERR ZERO FLG	ERR PRF FLG
0xFFF87020 FUNC_ERR_ADD Page 257	UNC_ERR_ADD[31:16]															
	UNC_ERR_ADD[15:3]														00	
0xFFF87024 FEDACSDIS Page 258	BankID1_inverse			reserved	SectorID1_inverse				BankID1			reserved	SectorID1			
	BankID0_inverse			reserved	SectorID0_inverse				BankID0			reserved	SectorID0			
0xFFF87028 - 0xFFF8702C	Reserved															
	Reserved															
0xFFF87030 FBPROT Page 259	Reserved															
																PROT L1DIS
0xFFF87034 FBSE Page 260	Reserved															
	BSE[15:0]															
0xFFF87038	Reserved															
	Reserved								Reserved for Flash API							
0xFFF8703C FBAC Page 261	reserved								OTPPROTDIS[7:0]							
	BAGP[7:0]								VREADST[7:0]							
0xFFF87040 FBFALLBACK Page 262	Reserved															
	Reserved												BANKPWR1 [1:0]		BANKPWR0 [1:0]	

Figure 7-6. Flash Control Register Summary (Continued)

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
0xFFF87044 FBPRDY Page 263	Reserved															
	PUMP RDY	Reserved													BANKRDY[3:0]	
0xFFF87048 FPAC1 Page 264	reserved						PSLEEP[10:0]									
	Reserved															PUMP PWR
0xFFF8704C FPAC2 Page 265	Reserved															
	PAGP[15:0]															
0xFFF87050 FMAC Page 266	Reserved															
	Reserved															BANK[2:0]
0xFFF87054	Reserved															
	Reserved for Flash API															
0xFFF87058 FEMU_DMSW Page 267	EMU_DMW[63:48]															
	EMU_DMW[47:32]															
0xFFF8705C FEMU_DLSW Page 268	EMU_DLW[31:16]															
	EMU_DLW[15:0]															
0xFFF87060 FEMU_ECC Page 269	Reserved															
	Reserved								EMU_ECC[7:0]							

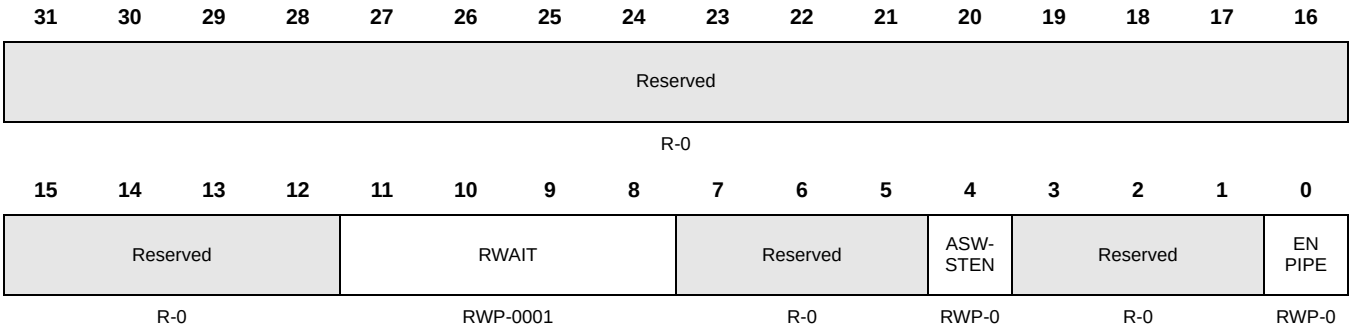
Figure 7-6. Flash Control Register Summary (Continued)

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
0xFFF87064	Reserved															
	Reserved for Flash API															
0xFFF87068 - 0xFFF8707C	Reserved															
	Reserved															
0xFFF87080 - 0xFFF870BC	Reserved															
	Reserved															
0xFFF870C0	BankID3_inverse			reserved	SectorID3_inverse				BankID3			reserved	SectorID3			
FEDACSDIS2 Page 270	BankID2_inverse			reserved	SectorID2_inverse				BankID2			reserved	SectorID2			

7.5.1 Flash Option Control Register (FRDCNTL)

FRDCNTL supports pipeline mode. There is only one FRDCNTL register for the entire F035 Flash.

Figure 7-7. Flash Option Control Register (FRDCNTL) [0xFFF87000]



-n = Value after reset, R=Read, WP=Write in Privilege Mode

Table 7-4. Flash Option Control Register (FRDCNTL) Field Descriptions

Bit	Name	Value	Description
31–12	Reserved		Reads return zeros and writes have no effect.
11–8	RWAIT	0x0–0xF	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 they are set to 0. **Note:** The required wait states for each HCLK frequency can be found in device datasheet.
7–5	Reserved		Reads return zeros and 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 device datasheet.
3–1	Reserved		Reads return zeros and writes have no effect.
0	ENPIPE		Enable Pipeline Mode
		0	Pipeline mode is disabled.
		1	Pipeline mode is enabled.

7.5.2 Flash Special Read Control Register (FSPRD)

Figure 7-8. Flash Special Read Control Register (FSPRD - 0xFFF87004)

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														RM1	RM0
R-0														RWP-0x0	

-*n* = Value after reset, R=Read, WP=Write in Privilege Mode

Table 7-5. Flash Special Read Control Register (FSPRD) Field Descriptions

Bit	Name	Value	Description
31–2	Reserved		Reads return zeros and writes have no effect.
1	RM1		Read Margin 1
		0	Read Margin 1 mode is enabled.
		1	Read Margin 1 mode is enabled.
0	RM0		Read Margin 0
		0	Read Margin 0 mode is disabled.
		1	Read Margin 0 mode is enabled.
			Note: If both RM0 and RM1 are set then Read Margin 0 is taken.

7.5.3 Flash Error Detection and Correction Control Register 1 (FEDACCTRL1 - 0xFFF87008)

Figure 7-9. Flash Error Detection and Correction Control Register 1 (FEDACCTRL1 - 0xFFF87008)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved							SUSP_IGNR	Reserved				EDACMODE			
R-0							RWP-0	R-0				RWP-0xA			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					EOF EN	EZF EN	EPEN	Reserved		EOCV	EZCV	EDACEN			
R-0						RWP-0	RWP-0	R-0		RWP-0	RWP-0	RWP-0x5			

-*n* = Value after reset, R=Read, WP=Write in Privilege Mode

This register controls ECC detection.

Table 7-6. Flash Error Detection and Correction Control Register 1 (FEDACCTRL1) Field Descriptions

Bit	Name	Value	Description
31–25	Reserved		Reads return zeros and 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 FWM.
		0	CPU suspend signal blocks error bits setting and un-freezing. The flash module blocks all errors from setting the error bits in emulation mode and blocks the un-freezing of the bits and registers by reading the FUNC_ERR_ADD register.
		1	CPU suspend has no effect on error bit setting and un-freezing. 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 un-freeze the error bits and other registers by reading the FUNC_ERR_ADD register even in emulation mode.
23–20	Reserved		Reads return zeros and writes have no effect.
19–16	EDACMODE		Error Correction Mode. In detection only mode any multi-bit or single bit error will result in an ECC_MUL_ERR and UERR will trigger. Single bit error flags, and profiling are disabled.
		0101	Error Detection mode. In this mode, only detection is enabled.
		all other values	Error Correction mode. In this mode, single bit error will be corrected if the ECC logic is enabled. Note: It is recommended writing "1010" to enable EDACMODE to guard against soft error from flipping EDACMODE to a disable state.

Table 7-6. Flash Error Detection and Correction Control Register 1 (FEDACCTRL1) Field Descriptions

Bit	Name	Value	Description
15–9	Reserved		Reads return zeros and writes have no effect.
10	EOFEN		Error on One Fail Enable When this bit is set, correctable errors where a one reads as a zero will generate an interrupt. Sector erase is required to restore a bit from zero to one.
		0	Error on One Fail interrupt is disabled.
		1	Error on One Fail interrupt is enabled.
9	EZFEN		Error on Zero Fail Enable When this bit is set, correctable errors where a zero bit reads as a one will generate an interrupt. This bit is used in systems where the user wants to restore failing zeros by reprogramming the failing bit. Failing ones can not be restored.
		0	Error on Zero Fail interrupt is disabled.
		1	Error on Zero Fail interrupt is enabled.
8	EPEN		Error Profiling Enable.
		0	Error profiling is disabled.
		1	Error profiling is enabled. The correctable error interrupt is generated when number of correctable bit errors detected and corrected has reached the threshold value defined in the FEDACCTRL2 register
7–6	Reserved		Reads return zeros and writes have no effect.
5	EOCV		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 ones) will generate ECC errors. The error counter for profiling will increment if all ones are detected.
		1	One condition valid is enabled. Reading of an erased location (64 data bits and the corresponding 8 ECC bits are all ones) will NOT generate ECC errors. The error counter for profiling will NOT increment if all ones are detected.
4	EZCV		Zero Condition Valid.
		0	Zero condition valid is disabled. Reading of all zeros (64 data bits and the corresponding 8 ECC bits are all zeros) will generate ECC errors. The error counter for profiling will increment if all zeros are detected.
		1	Zero condition valid is enabled. Reading of all zeros (64 data bits and the corresponding 8 ECC bits are all zeros) will NOT generate ECC errors. The error counter for profiling will NOT increment if all zeros are detected.

Table 7-6. Flash Error Detection and Correction Control Register 1 (FEDACCTRL1) Field Descriptions

Bit	Name	Value	Description
3–0	EDACEN		Error Detection and Correction Enable
		0101	Disable - Error Detection and Correction is disabled
		all others	Enable - Error Detection and Correction is enabled
			Note: It is recommended writing “1010” to enable EDACEN to guard against soft error from flipping EDACEN to a disable state.
			Error Detection and Correction is enabled by default.

7.5.4 Flash Error Correction and Correction Control Register 2 (FEDACCTRL2 - 0xFFF8700C)

Figure 7-10. Flash Error Correction and Correction Control Register 2 (FEDACCTRL2 - 0xFFF8700C)

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC_THRESHOLD															
RWP-0															

-n = Value after reset, R=Read, WP=Write in Privilege Mode

Table 7-7. Flash Error Correction Control and Correction Register 2 (FEDACCTRL2) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Reads return zeros and writes have no effect.
15–0	SEC_THRESHOLD		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.

7.5.5 Flash Error Correction Counter Register (FCOR_ERR_CNT - 0xFFF87010)

Figure 7-11. Flash Error Correction Counter Register (FCOR_ERR_CNT - 0xFFF87010)

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COR_ERR_CNT															
RWP-0															

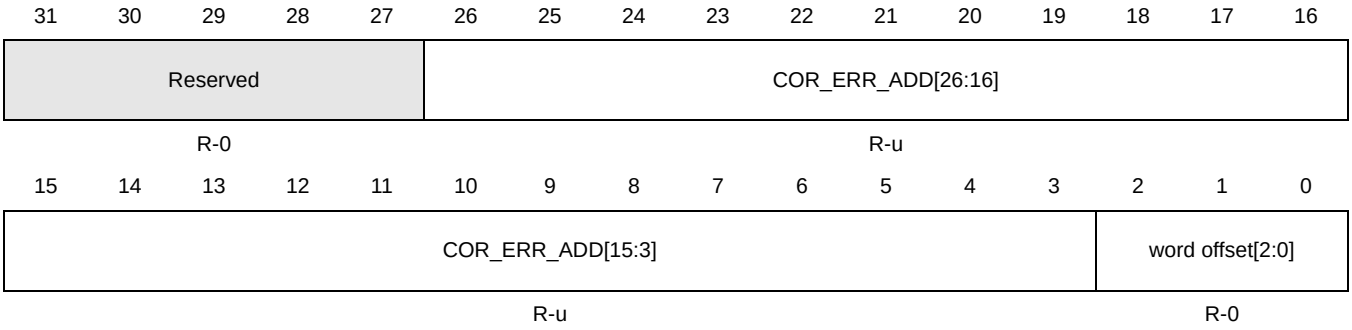
-n = Value after reset, R=Read, WP=Write in Privilege Mode

Table 7-8. Flash Error Correction Counter Register (FCOR_ERR_CNT) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Reads return zeros and writes have no effect.
15–0	COR_ERR_CNT		Correctable Error Counter This 16 bit counter contains the number of correctable error occurrences. A write to this register with any value will reset the counter to all zeros. The counter resets to 0 when it is equal to the threshold value and continues to increment if it detects an error again. This counter is frozen in emulation mode. The register only counts when the error profiling bit EPEN in the FEDACCTRL1 register is set to one. **Note:** Clear the this register before changing the SEC_THRESHOLD field in the FEDACCTRL2 register.

7.5.6 Flash Correctable Error Address (FCOR_ERR_ADD - 0xFFF87014)

Figure 7-12. Flash Correctable Error Address (FCOR_ERR_ADD - 0xFFF87014)



-n = Value after reset, R=Read, WP=Write in Privilege Mode
-u = unchanged value on internal reset, cleared on power up

This register is not changed after the nRST. This register will not get updated when either ERR_ZERO_FLG or ERR_ONE_FLG is '1'. Also, the FCOR_ERR_POS register will not get updated when either bit is set.

Note: 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 an correctable error is detected.

Table 7-9. Flash Correctable Error Address (FCOR_ERR_ADD) Field Descriptions

Bit	Name	Value	Description
31–27	Reserved		Reads return zeros and writes have no effect.
26–3	COR_ERR_ADD		Error Address COR_ERR_ADD records the CPU logical address of which a correctable error (ECC single bit error) is detected by the ECC logic in the CPU. This register is frozen from changing during emulation mode.
2–0	word offset	0	The last 3 digit of the correctable error address. Since ECC is checked on 64 bit data, the address captured is aligned to a 64-bit boundary with bit[2:0] tied to 0. These bits always read 0; writes have no effect

7.5.7 Correctable Error Position Register (FCOR_ERR_POS)

This register is not changed with the nRST and contains unknown data at powerup. This register will **not** get updated when either ERR_ZERO_FLG or ERR_ONE_FLG is '1'. Also, the FCOR_ERR_ADD register will not get updated when either bit is set.

Figure 7-13. *Correctable Error Position Register (FCOR_ERR_POS) [offset = 0xFFF87018]*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				Reserved				Reserved				Reserved			
R-0				R-0101				R-0				R-0101			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							ECC ERR	SERR_POS[7:0]							
R-0							R-u	R-u							

R=Read, WP=Write in privilege mode,-*n* = Value after reset, R=Read, WP=Write in Privilege Mode
-u = unchanged value on internal reset, cleared on power up

.

Table 7-10. Correctable Error Address Register (FCOR_ERR_ADD) Field Descriptions

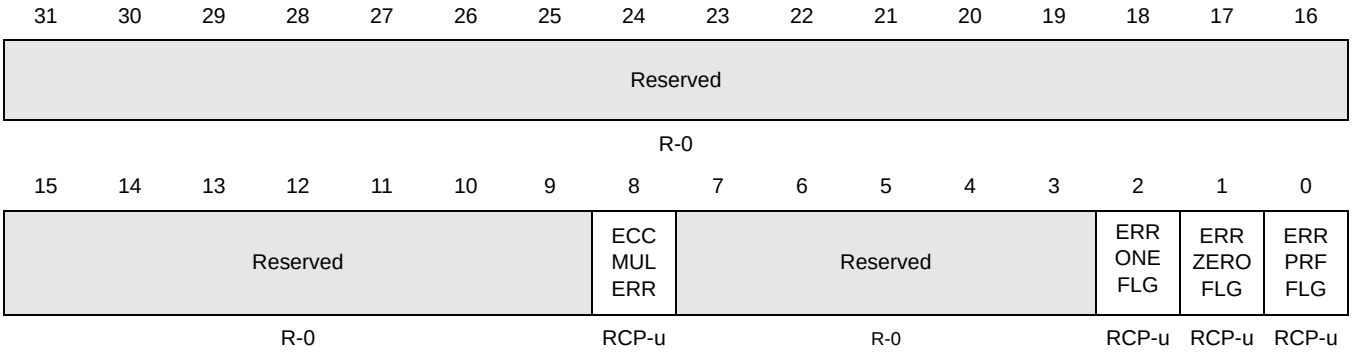
Bit	Name	Value	Description
31–28	Reserved	0000	Reads return zeros and writes have no effect.
27–24	Reserved	0101	Reads return 0x5 and do NOT write to this field.
23–20	Reserved	0000	Reads return zeros and writes have no effect.
19–16	Reserved	0101	Reads return 0x5 and do NOT write to this field.
8	ECC_ERR		Error Type This bit indicates whether the single error detected is a data bit error or check bit error. The error is not a check bit error The error is a check bit error

Table 7-10. Correctable Error Address Register (FCOR_ERR_ADD) Field Descriptions (Continued)

Bit	Name	Value			Description
7-0	SERR_POS				Single Error Position SERRPOSITION records the binary encoded error position of which a single error is detected. Error position is captured into SERRPOSITION when a single bit error is detected. Following examples illustrate the error position being captured in SERRPOSITION. This register is frozen from changing during suspend mode.
		FSERRPOSTION			Comments
		ECC_ERR	7:0		
		0	0x00	Data Bit 0	Values 0-63 is data
		0	0x07	Data Bit 7	
		0	0x3F	Data Bit 63	
		1	0x00	ECC Bit 0	Bit 8, ECC_ERR indicates ECC bits
		1	0x07	ECC Bit 7	

7.5.8 Flash Error Status Register (FEDACSTATUS - 0xFFF8701C)

Figure 7-14. Flash Error Status Register (FEDACSTATUS - 0xFFF8701C)



-n = Value after reset, R=Read, CP=Clear in Privilege Mode by writing a 1,
-u = unchanged value on internal reset, cleared on power up

All these error status bits can be cleared by writing a one to the bit. Writing a zero has no effect.

These error bits are not set in the emulation mode but they can be cleared in the emulation mode by writing '1's to the bits. By setting the SUSP_IGNR bit to '1' these error bits can be set in suspend mode.

Bits 0 to 2 show correctable errors while bits 8 show uncorrectable errors.

When the correctable errors are detected, the current address is stored in the FCOR_ERR_ADD register and frozen.The correctable errors in bits 2:0 must be cleared before the end of the interrupt service routine or else the interrupt will re-issue. A one in bits 1 or 2 will prevent bits 1 or 2 from setting. Bit 1 will not set if bit 2 is set. The FCOR_ERR_ADD and FCOR_ERR_POS registers will not update while bit 1 or 2 are set.

When the uncorrectable errors are detected, the current address is stored in the FUNC_ERR_ADD register and frozen. The FUNC_ERR_ADD will not change again unless it is first unfrozen by being read. Additional uncorrectable errors are blocked from setting additional error bits until the FUNC_ERR_ADD is unfrozen. All the uncorrectable error bits will assert a UERR to the ESM module while unfrozen. Then, the UERR signal is frozen until the FUNC_ERR_ADD is read.

Flash wrapper has higher priority than the CPU to set the error bit. This could cause a condition where the error bit is cleared before it can be read by the CPU. To avoid the freezing of the error bits and error address under this situation, the application should read the FUNC_ERR_ADD to un-freeze the error bits and error address if an interrupt is generated but no error bit is set.

Table 7-11. Flash Error Status Register (FEDACSTATUS) Field Descriptions

Bit	Name	Value	Description
31–9	Reserved		Reads return zeros and writes have no effect.
8	ECC MUL ERR		Multiple bit ECC Status Flag
		0	Un-correctable double bit or multiple bit ECC Error is not detected
		1	Un-correctable double bit or multiple bit ECC Error is detected
			Note: Address error in flash bank is also treated as multiple bit ECC Error.

Bit	Name	Value	Description
7–3	Reserved		Reads return zeros and writes have no effect.
2	ERR_ONE_FLG		Error On One Fail Status Flag If EOFEN (Error on One Fail Enable) bit is set then ERR_ONE_FLG flag is set when a correctable error is detected and the bit at fault is read as 0 and corrected to be 1.
		0	Error on One Fail is not detected.
		1	Error on One Fail is detected.
			This bit will not get set when either ERR_ZERO_FLG or ERR_ONE_FLG is '1'. This also means the FCOR_ERR_POS and FCOR_ERR_ADD will not get updated when either bit is set.
1	ERR_ZERO_FLG		Error On Zero Fail Status Flag If EZFEN (Error on Zero Fail Enable) bit is set then ERR_ZERO_FLG flag is set when a correctable error is detected and the bit at fault is read as 1 and corrected to be 0.
		0	Error on Zero Fail is not detected.
		1	Error on Zero Fail is detected.
			Note: This bit will not get set when either ERR_ZERO_FLG or ERR_ONE_FLG is '1'. This also means the FCOR_ERR_POS and FCOR_ERR_ADDR will not get updated when either bit is set.
0	ERR PRF FLG		ERR_PRF_FLG flag. Only apply when EPEN is set.
		0	Profiling error is not detected.
		1	Profiling error is detected. The number of occurrences of correctable error detected and corrected by ECC logic in the CPU has reached the programmed threshold value stored in FEDACCTRL2 register

7.5.9 Flash Un-correctable Error Address (FUNC_ERR_ADD - 0xFFF87020)

Figure 7-15. Flash Un-correctable Error Address (FUNC_ERR_ADD - 0xFFF87020)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UNC_ERR_ADD[31:16]															
R-u															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UNC_ERR_ADD[15:3]													00		
R-u													R-0		

-n = Value after reset, R=Read, WP=Write in Privilege Mode
-u = unchanged value on internal reset, cleared on power up

During emulation mode, this address is frozen even when read. By setting the SUSP_IGNR bit, this register can be un-frozen in emulation mode.

This register is not changed with the reset signal and contains unknown data at powerup.

Table 7-12. Flash Un-correctable Error Address (FUNC_ERR_ADD) Field Descriptions

Bit	Name	Value	Description
31–3	UNC_ERR_ADD		Un-correctable Error Address UNC_ERR_ADD records the CPU logical address of which an un-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. It only captures address of 22:3 for multiple bit ECC errors.
2–0	word offset	0	The last 3 digit of the address. Since ECC is checked on 64 bit data, the address captured is aligned to a 64-bit boundary with bit[2:0] tied to 0. If the ECC error was due to an address bit then this value will be questionable.

7.5.10 Flash Error Detection Sector Disable (FEDACSDIS - 0xFFF87024)

Figure 7-16. Flash Error Detection Sector Disable (FEDACSDIS - 0xFFF87024)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BankID1_inverse			reserved	SectorID1_inverse				BankID1			reserved	SectorID1			
RWP-0			R-0	RWP-0				RWP-0			R-0	RWP-0			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
BankID0_inverse			reserved	SectorID0_inverse				BankID0			reserved	SectorID0			
RWP-0			R-0	RWP-0				RWP-0			R-0	RWP-0			

-n = Value after reset, R=Read, WP=Write in Privilege Mode

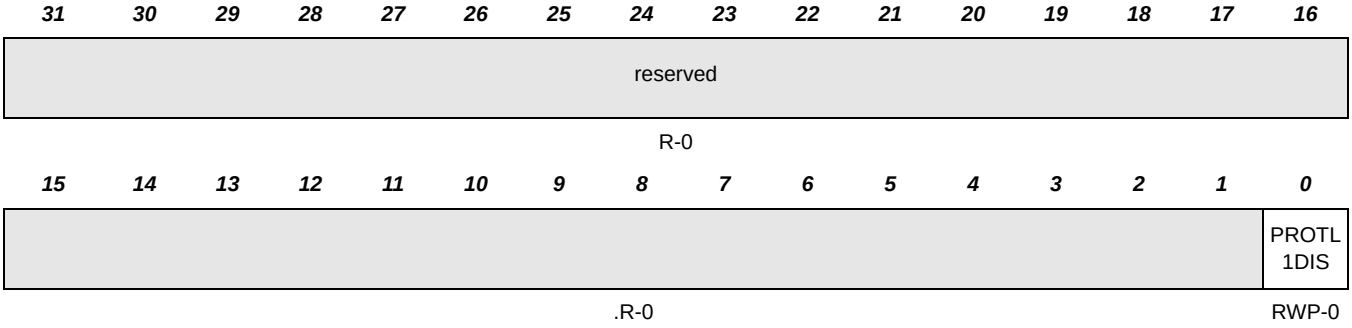
The sectors specified in this register is excluded from error detection checking. The user must specify the bank/sector to be excluded and its inverse value in this register. Only when the programmed bank/sector ID value and its calculated inverted value matches the programmed inverse value will the sector selected be excluded from ECC checking. FEDACSDIS2 serves the same purpose. Bank/sector ID is the Bank NO./ Sector NO. in the device datasheet.

Table 7-13. Flash Error Detection Sector Disable (FEDACSDIS) Field Descriptions

Bit	Name	Value	Description
31–29	BankID1_Inverse		This is the inverse value of the bank ID which contains the sector to be disabled from error detection checking.
28	Reserved		Reads return zeros and writes have no effect.
27–24	SectorID1_Inverse		This is the inverse value of the sector ID to be disabled from error detection checking.
23–21	BankID1		This is the value of the bank ID which contains the sector to be disabled from error detection checking.
20	Reserved		Reads return zeros and writes have no effect.
19–16	Sector1D1		This is the value of the sector ID to be disabled from error detection checking.
15–13	BankID0_Inverse		This is the inverse value of the bank ID which contains the sector to be disabled from error detection checking.
12	Reserved		Reads return zeros and writes have no effect.
11–8	SectorID0_Inverse		This is the inverse value of the sector ID to be disabled from error detection checking.
7–5	BankID0		This is the value of the bank ID which contains the sector to be disabled from error detection checking.
4	Reserved		Reads return zeros and writes have no effect.
3–0	Sector1D0		This is the value of the sector ID to be disabled from error detection checking.

7.5.11 Flash Bank Protection Register (FBPROT - 0xFFF87030)

Figure 7-17. Flash Bank Protection Register (FBPROT - 0xFFF87030)



-n = Value after reset, R=Read, WP=Write in Privilege Mode

Table 7-14. Flash Bank Protection Register (FBPROT) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved		Reads return zeros and 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 disabled.
		1	Level 1 protection is enabled.

7.5.12 Flash Bank Sector Enable Register (FBSE- 0xFFF87034)

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. As only one bank at a time can be selected by FMAC, only the register for the bank selected appears at this address.

Figure 7-18. Flash Bank Sector Enable Register (FBSE - 0x34)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
BSE[15:0]															
RWP-0															

-*n* = Value after reset, R = Read, WP = Write in Privilege Mode

.

Table 7-15. Flash Bank Sector Enable Register (FBSE)

Bit	Name	Value	Description
31–16	Reserved		Reads return zeros and writes have no effect.
15-0	BSE[15:0]		Bank Sector Enable Each bit corresponds to a flash sector in the bank specified by the FMAC register. This bit 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.

7.5.13 Flash Bank Access Control Register (FBAC - 0x3C)

Figure 7-19. Flash Bank Access Control Register (FBAC - 0xFFF8703C)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved								OTPPROTDIS[7:0]							
R-0								RWP-00000000							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
BAGP[7:0]								VREADST[7:0]							
RWP-0								RWP-00001111							

-*n* = Value after reset, R = Read, WP = Write in Privilege Mode

Table 7-16. Flash Bank Access Control Register (FBAC) Field Descriptions

Bit	Name	Value	Description
31–24	Reserved		Reads return zeros and writes have no effect.
23-16	OTPPROTDIS[7:0]		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[7:0]		Bank Active Grace Period. These bits contain the starting count value for the BAGP down counter. Any access to a given bank causes its BAGP counter to reload the BAGP value for that bank. After the last access to this flash bank, the down counter delays from 0 to 255 prescaled 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[7:0]		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.

7.5.14 Flash Bank Fallback Power Register (FBFALLBACK - 0xFFF87040)

Figure 7-20. Flash Bank Fallback Power Register (FBFALLBACK - 0xFFF87040)

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												BANKPWR1 [1:0]		BANKPWR0 [1:0]	
RWP-0xF												RWP-11		RWP-11	

-*n* = Value after reset, R = Read, WP = Write in Privilege Mode

Table 7-17. Flash Bank Fallback Power Register (FBFALLBACK) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Reads return zeros and writes have no effect.
15–8	Reserved		Not used in this device.
3-2	BANKPWR1[1:0]		Bank 1 Fallback Power Mode
		00	Bank sleep mode
		01	Bank standby mode
		10	Reserved
		11	Bank active mode
1-0	BANKPWR0[1:0]		Bank 0 Fallback Power Mode - See BANKPWR1[1:0] for details.

7.5.15 Flash Bank/Pump Ready Register (FBPRDY - 0xFFF87044)

FBRDY allows the user to determine if the associated bank or the pump is ready for read access.

Figure 7-21. Flash Bank/Pump Ready Register (FBPRDY - 0xFFF87044)

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUMP RDY	Reserved							Reserved						BANKRDY[1:0]	
R-1	R-0							R-0x3F						R-0x3	

-*n* = Value after reset, R = Read, WP = Write in Privilege Mode

.

Table 7-18. Flash Bank/Pump Ready Register (FBPRDY) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Reads return zeros and writes have no effect.
15	PUMPRDY		Pump Ready Pump Ready is a read-only bit which allows software to determine if the pump is ready for flash access before attempting the actual access. If an access is made to a bank when the pump is not ready, wait states are asserted until it becomes ready.
		0	Pump is not ready
		1	Pump is ready, in active power state.
14-8	Reserved		Reads return zeros and writes have no effect.
7-2	Reserved		Reads return Ones and writes have no effect.
1-0	BANKRDY[1:0]		Bank Ready This is a read-only register which allows software to determine if the corresponding bank is ready for flash access before the access is attempted. **Note:** User should wait for both the pump and the targeted bank to be ready before attempting an access.
		0	The corresponding bank is not ready. For example, BANKRDY[1]=0 means BANK1 is not ready for access.
		1	The corresponding bank is ready, in active power mode. For example, BANKRDY[1]=1 means BANK1 is ready for access.

7.5.16 Flash Pump Access Control Register 1 (FPAC1 - 0xFFF87048)

Figure 7-22. Flash Pump Access Control Register 1 (FPAC1 - 0xFFF87048)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved					PSLEEP[10:0]										
R-0					RWP-0xC8										

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															PUMP PWR
R-0															RWP-1

-*n* = Value after reset, R = Read, WP = Write in Privilege Mode

.

Table 7-19. Flash Pump Access Control Register 1 (FPAC1) Field Descriptions

Bit	Name	Value	Description
31–27	Reserved		Reads return zeros and writes have no effect.
26-16	PSLEEP[10:0]		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		Reads return zeros and writes have no effect.
0	PUMPPWR		Flash Charge Pump Fallback Power Mode
		0	Sleep (all pump circuits disabled)
		1	Active (all pump circuits active)

7.5.17 Flash Pump Access Control Register 2 (FPAC2 - 0xFFF8704C)

Figure 7-23. Flash Pump Access Control Register 2 (FPAC2 - 0xFFF8704C)

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PAGP[15:0]															
RWP-0															

-*n* = Value after reset, R = Read, WP = Write in Privilege Mode

Table 7-20. Flash Pump Access Control Register 2 (FPAC2) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Reads return zeros and writes have no effect.
15-0	PAGP[15:0]		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.

7.5.18 Flash Module Access Control Register (FMAC - 0xFFF87050)

Figure 7-24. Flash Module Access Control Register (FMAC - 0xFFF87050)

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													BANK[2:0]		
R-0													RWP-0		

-*n* = Value after reset, R = Read, WP = Write in Privilege Mode

Table 7-21. Flash Module Access Control Register (FMAC) Field Descriptions

Bit	Name	Value	Description
31–3	Reserved		Reads return zeros and writes have no effect.
2-0	BANK[2:0]		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 less than 8 banks are configured and if BANK[2:0] is selected for an un-implemented bank then the BANK[2:0] will set itself to the highest implemented bank. To determine the number of implemented banks, write “111” to this register and then read it back for the number of implemented banks -1.

7.5.18.1 EEPROM Emulation Data MSW Register (FEMU_DMSW)

Figure 7-25. EEPROM Emulation Data MSW Register (FEMU_DMSW) [offset = 0xFFF87058]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EMU_DMW[63:48]															
RWP-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EMU_DMW[47:32]															
RWP-0															

-n = Value after reset, R=Read, WP=Write in Privilege Mode

Table 7-22. *EEPROM Emulation Data MSW Register (FEMU_DMSW)* Field Descriptions

Bit	Name	Value	Description
31-0	EMU_DMW		EEPROM Emulation Most Significant Data Word. This register contains the upper 32 bits of the 64-bit EEPROM emulation data word. When this register is written, the corresponding ECC word is updated in FEMU_ECC register.

7.5.18.2 EEPROM Emulation Data LSW Register (FEMU_DLSW)

Figure 7-26. EEPROM Emulation Data LSW Register (FEMU_DLSW) [offset = 0xFFF8705C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EMU_DLW[31:16]															
RWP-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EMU_DLW[15:0]															
RWP-0															

-n = Value after reset, R=Read, WP=Write in Privilege Mode

Table 7-23. Field Descriptions

Bit	Name	Value	Description
31-0	EMU_DLW		EEPROM Emulation Least Significant Data Word. This register contains the lower 32 bits of the 64-bit EEPROM emulation data word. When this register is written, the corresponding ECC word is updated in FEMU_ECC register.

7.5.19 Flash Emulation ECC Register (FEMU_ECC - 0xFFF87060)

Figure 7-27. Flash Emulation ECC Register (FEMU_ECC - 0xFFF87060)

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								EMU_ECC[7:0]							
R-0								RWP-00000011 †							

-n = Value after reset, R=Read, WP=Write in Privilege Mode; †-see text

Table 7-24. Flash Emulation ECC Register (FEMU_ECC) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return zeros and writes have no effect.
7-0	EMU_ECC		This register can be written by the CPU in any mode. When either FEMU_DMSW or FEMU_DLSW register is written, the flash wrapper will return the correct ECC value to this register. The FEMU_ADDR register is also used in the calculation when address is a part of the ECC calculation and the address must be written before the last FEMU_DxSW value.

7.5.20 Flash Error Detection Sector Disable (FEDACSDIS2 - 0xFFF870C0)

Figure 7-28. Flash Error Detection Sector Disable (FEDACSDIS2 - 0xFFF870C0)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BankID3_inverse			reserved	SectorID3_inverse				BankID3			reserved	SectorID3			
RWP-0			R-0	RWP-0				RWP-0			R-0	RWP-0			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
BankID2_inverse			reserved	SectorID2_inverse				BankID2			reserved	SectorID2			
RWP-0			R-0	RWP-0				RWP-0			R-0	RWP-0			

-n = Value after reset, R=Read, WP=Write in Privilege Mode

The sectors specified in this register is excluded from error detection checking. The user must specify the bank/sector to be excluded and its inverse value in this register. The user must specify the bank/sector to be excluded and its inverse value in this register. Only when the programmed bank/sector ID value and its calculated inverted value matches the programmed inverse value will the sector selected be excluded from ECC checking errors. FEDACSDIS serves the same purpose.

Table 7-25. Flash Error Detection Sector Disable (FEDACSDIS2) Field Descriptions

Bit	Name	Value	Description
31–29	BankID3_Inverse		This is the inverse value of the bank ID which contains the sector to be disabled from error detection checking.
28	Reserved		Reads return zeros and writes have no effect.
27–24	SectorID3_Inverse		This is the inverse value of the sector ID to be disabled from error detection checking.
23–21	BankID3		This is the value of the bank ID which contains the sector to be disabled from error detection checking.
20	Reserved		Reads return zeros and writes have no effect.
19–16	Sector1D3		This is the value of the sector ID to be disabled from error detection checking.
15–13	BankID2_Inverse		This is the inverse value of the bank ID which contains the sector to be disabled from error detection checking.
12	Reserved		Reads return zeros and writes have no effect.
11–8	SectorID2_Inverse		This is the inverse value of the sector ID to be disabled from error detection checking.
7–5	BankID2		This is the value of the bank ID which contains the sector to be disabled from error detection checking.
4	Reserved		Reads return zeros and writes have no effect.
3–0	Sector1D2		This is the value of the sector ID to be disabled from error detection checking.

Cortex-M3 Vectored Interrupt Manager (M3VIM) Module

This chapter describes the behavior of the Cortex M3 vectored interrupt manager (M3VIM) module of the TMS470M Series of microcontrollers.

Topic **Page**

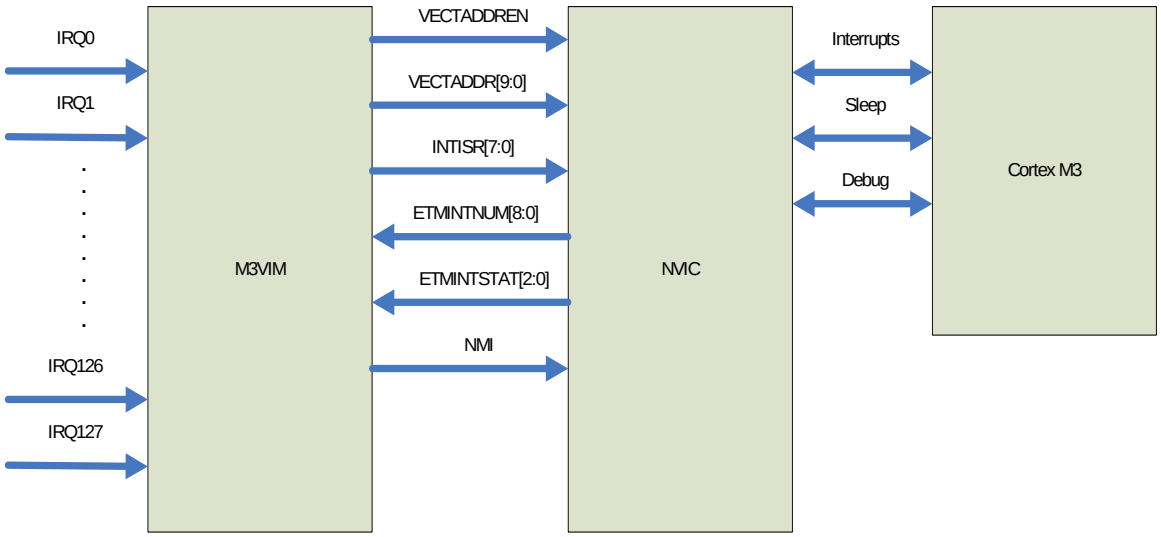
8.1 Overview 272
8.2 Interrupt management 274
8.3 M3VIM Operation 279
8.4 Capture event sources 281
8.5 Programmer's Model Notes 282
8.6 Registers 284

8.1 Overview

The TMS470M Series interrupt architecture for Cortex M3 includes a vectored interrupt manager (M3VIM) that provides hardware assistance for prioritizing and controlling the many interrupt sources present on a device. Interrupts are caused by events outside 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. The interrupt service routine is a small program designed to handle the event occurrence in a timely manner.

The Cortex M3 processor contains a system peripheral, the NVIC, which controls system exceptions and acts as a system control coprocessor. The M3VIM interfaces with the NVIC in order to control interrupts. A high level block diagram of the M3VIM connection to Cortex M3 is shown below.

Figure 8-1. Block Diagram



8.1.1 Interrupt Handling at the CPU

The ARM Cortex M3 CPU provides two vectors for interrupt requests—fast non-maskable interrupt requests (INTNMI) and normal vectorized interrupt requests (INTISR). The INTNMI is always enabled. INTISR vectored interrupts are disabled at reset and must be enabled by programming the NVIC module with a short initialization sequence. The INTNMI interrupt request has higher priority than the INTISR and is handled first.

When the CPU recognizes an interrupt request, the Cortex-M3 changes mode from thread to handler. After the interrupt is recognized by the CPU, the program counter jumps to the appropriate handler location found in the vector table.

Support for hardware vectored interrupts eliminates the need for a software interrupt handler to determine the source of an interrupt before branching to the interrupt service routine (ISR).

8.1.1.1 Non-Maskable Interrupt (INTNMI)

M3VIM channel 0 and channel 1 are called INTNMI channels and are non-maskable. These 2 channels receive INT0 and INT1 respectively and can not be changed.

8.1.2 Nesting Behavior

The Cortex M3 and M3VIM contain hardware support for nested interrupts Nesting is defined as the ability of a second, higher priority interrupt to interrupt the handling of a currently active interrupt.

The default behavior of the M3VIM at reset is to disable the nested interrupt behavior. When nesting is disabled, only the INTISR[0] autovectored interrupt is used to interrupt the core. In this mode, a currently executing ISR must complete before another ISR may execute, even if the second interrupt request has higher priority than the currently executing request. There is one exception to this rule - an INTNMI will interrupt the execution of an autovectored INTISR interrupt.

If nesting is enabled, the M3VIM will allow up to eight levels of nesting. The eight levels will be supported on the INTISR[7:0] signals.

The M3VIM will provide a nesting level status register byte, NEST_LEVEL, in the NEST_STAT register. Each bit in this register will be set when the corresponding nesting level has been reached. For example, INTISR[2]=1 will set the third bit of this register high. The register will maintain contents until cleared by privileged software. The purpose of this register is to allow some profiling of nesting levels by a software developer.

8.1.3 Interrupt Generation at the Peripheral

Interrupts begin 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 TMS470M Series peripherals are capable of requesting interrupts on more than one interrupt channel. The M3VIM supports both pulse and level sensitive interrupts from peripherals.

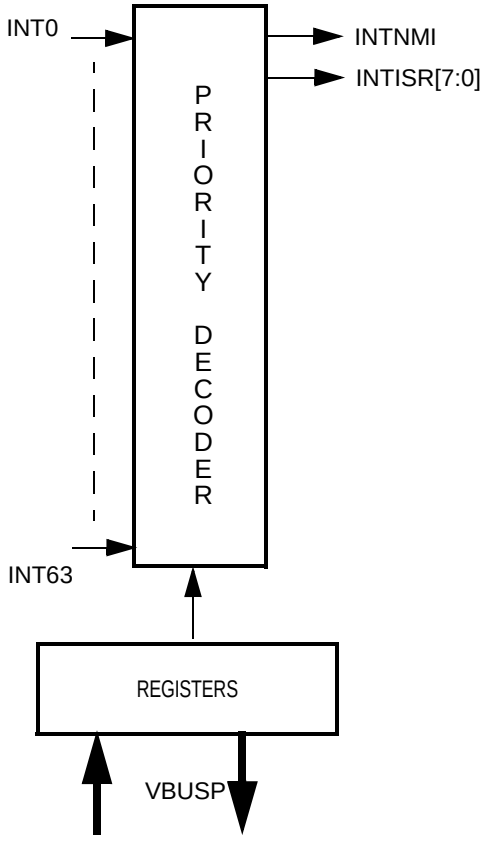
Interrupts are not always generated when an event occurs; the peripheral must make an interrupt request to the vectored interrupt manager based upon the event occurrence. Typically, the peripheral contains:

- An interrupt flag bit for each event to signify the event occurrence
    - o An interrupt enable bit to control whether the event occurrence causes an interrupt request to the M3VIM.

8.2 Interrupt management

A general block diagram of the vectored interrupt manager is shown in Figure 8-2

Figure 8-2. Vectored Interrupt Manager block diagram



The M3VIM supports 48 interrupt request lines. The peripheral interrupt requests are hardwired to each of the M3VIM interrupt request inputs, and ***these connections are device specific***. (See the device specific data sheet for details on which module is tied to which interrupt request channel.) All interrupt request inputs are mapped to a specific channel; this mapping is fully programmable and allows the software to re-order the interrupt priority. All requests pass through a synchronizer to prevent setup time violations; the M3VIM samples the interrupt requests from the synchronizer every peripheral clock (***vbus_clk***) cycle. The M3VIM combines the 48 channels into two outputs – an INTNMI request to the CPU and an INTISR vectored request to the CPU.

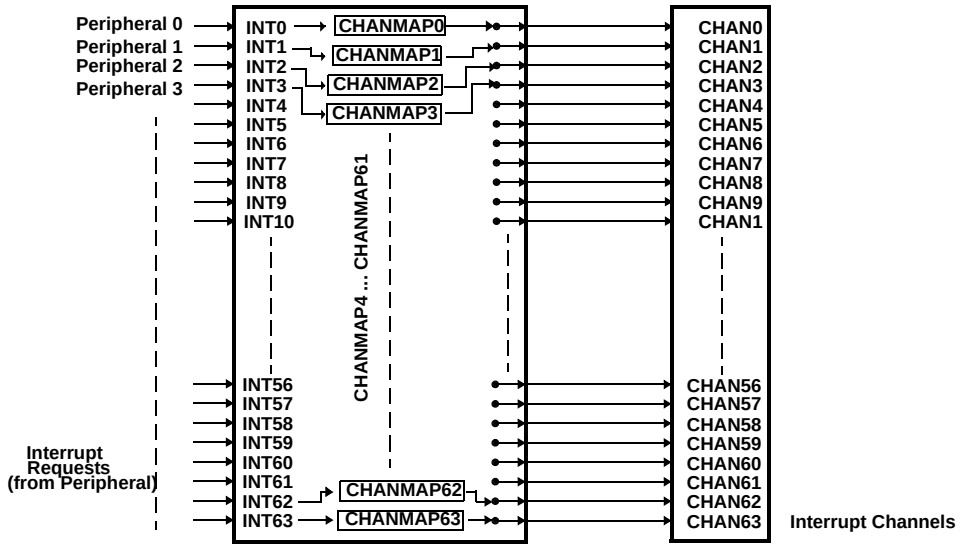
The M3VIM performs the following functions:

- Maps the 48 interrupt requests to the 48 interrupt channels
- Provides programmable priority through interrupt request mapping
- Manages the interrupt channels through masking
- Prioritizes the interrupt channels to the CPU
- Provides the CPU with the address of the ISR vector offset in the vector table for the activated INTISR

8.2.1 M3VIM *Interrupt request management*

The programmability of the M3VIM allows software to control the interrupt priority. A block diagram of the M3VIM interrupt request management, shown in the default state following reset, is provided in Figure 8-3. In the reset state, the M3VIM maps all of the 48 interrupts request in the system to the respective 48 interrupt channels.

Figure 8-3. M3VIM in Default State



As illustrated in Figure 8-3, the M3VIM is used to map the interrupt requests from peripheral modules to the interrupt channel priority.

For each interrupt channel, CHAN[x], there is a corresponding mapping register bit field CHANMAPx[6:0], which determines which M3VIM interrupt request the interrupt channel maps to. With this scheme, the same interrupt request can be map to multiple channels.

Note: INTNMI channel
The INT0 and INT1 interrupt request are fixed to channel 0 and channel 1 respectively.

Figure 8-4. M3VIM in a programmed State

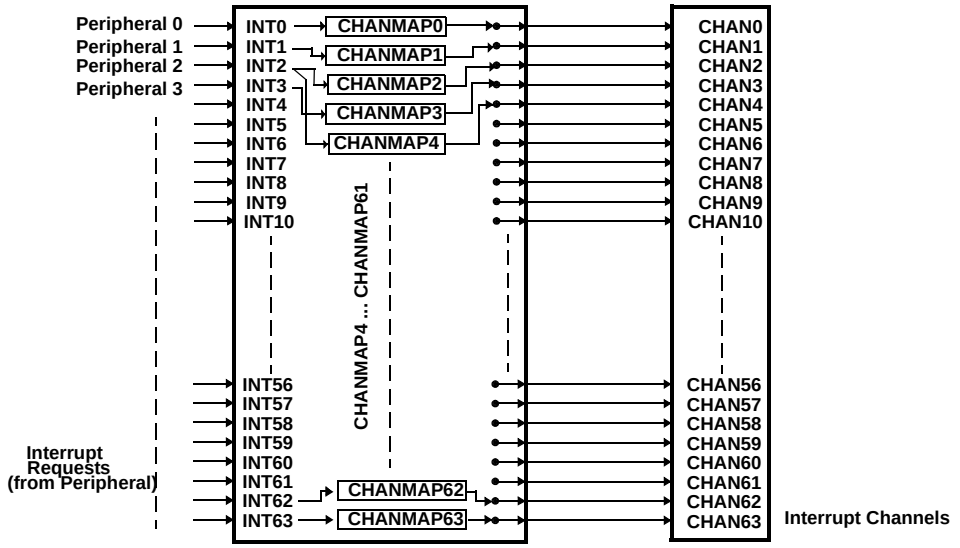


Figure 8-4 shows a partial programming of the M3VIM. INT2 is mapped to both Channel 2 and 4, INT3 is mapped to channel 3.

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 REQMASK register.

When channel 2 is enabled, the priority is:

a.INT0

b.INT1

c.INT2

d.INT3

e....

Disabling channel 2, the priority becomes:

a.INT0

b.INT1

c.INT3

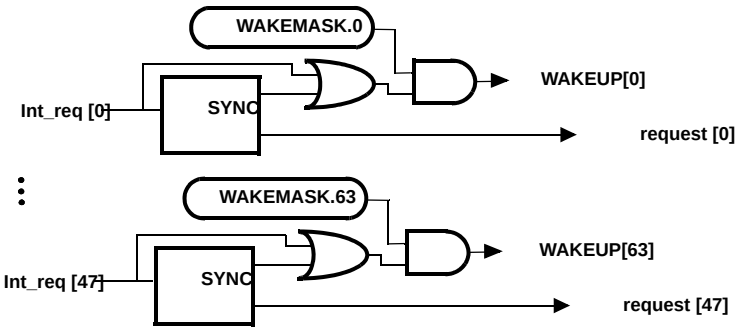
d.INT2

e....

8.2.2 M3VIM *Wake-up Interrupt*

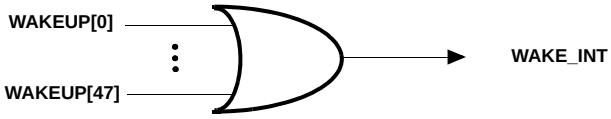
The wake-up interrupts are used to come out of the Low Power Mode (LPM). Any interrupt can be used to wake-up the device. After reset, all interrupts are set to wake-up from LPM. However, the M3VIM can mask unwanted interrupt line for wake-up by using the WAKEMASK register.

Figure 8-5. Detail of the Interrupt request input.



The Figure 8-5 shows the implementation of each interrupt request coming from the peripherals. The WAKEMASK registers will enable/disable an interrupt for wake-up from low-power mode. All wake-up interrupts are “ORed” into a single signal connected to the Global Clock Module (see Figure 8-6).

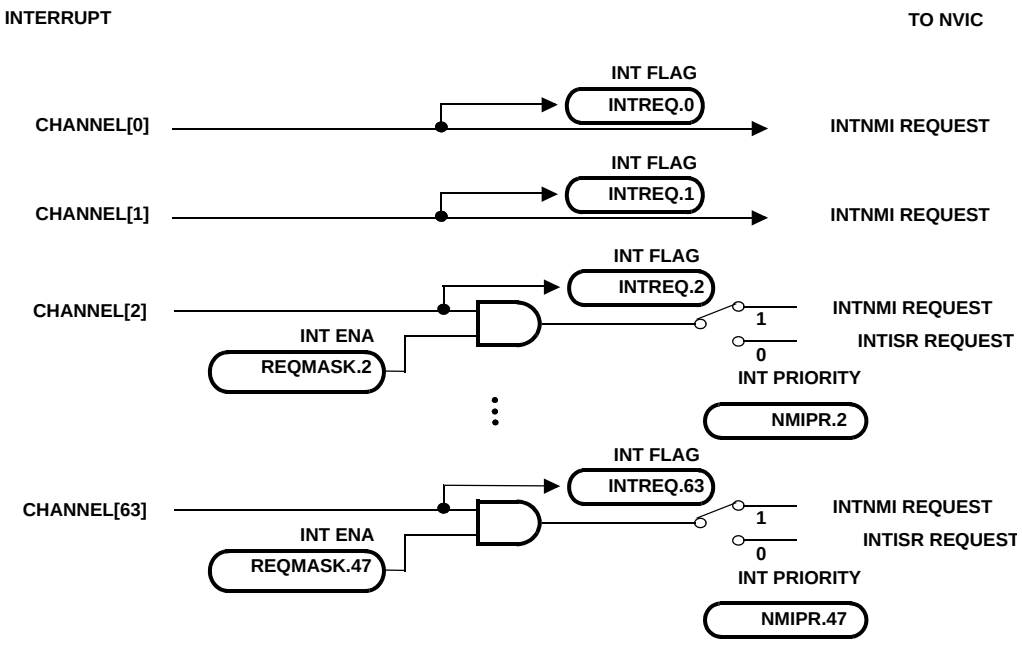
Figure 8-6. Wake-up interrupt generation



8.2.3 M3VIM *Input Channel Management*

On the input side, the M3VIM enables channels on a channel-by-channel basis (in the REQMASK registers); unused channels may be masked to prevent spurious interrupts. Each interrupt channel can be designated to send either an INTNMI or INTISR request to the CPU (in the NMIPR register).

Figure 8-7. Channel management block diagram

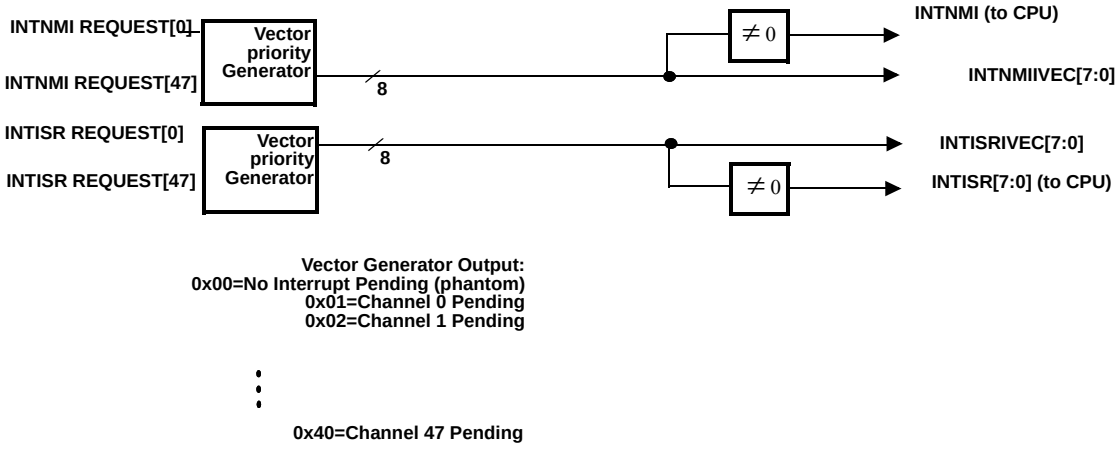


Note: INTNMI Channel
Channel 0 and channel 1 are not maskable by the REQMASK bit and both channel are routed exclusively to INTNMI request line (NMIPR.0 and NMIPR.1 have no effect).

8.2.4 M3VIM *Prioritization*

The M3VIM prioritizes the received interrupts based upon a programmed prioritization scheme. The M3VIM can send two interrupt requests to the CPU simultaneously—one INTISR and one INTNMI. The INTNMI has greater priority and is always handled first. The M3VIM provides a default prioritization scheme which sends the lowest numbered active channel (in each INTNMI and INTISR interrupt request) to the CPU. Within the INTNMI and INTISR classes of interrupts, the lowest channel has the highest priority interrupt when multiple interrupts of the same type are processed. However, the M3VIM provides a fully programmable priority scheme (request to channel mapping). Each request could be affected with a unique priority weight, which will allow the M3VIM to decide which request has the highest priority at a given time.

Figure 8-8. Priority, Vector and ISR address generation block diagram



Once the M3VIM has generated the vector corresponding to the highest active interrupt request, it updates the INTNMIIVEC or the INTISRIVEC register, depending on the class of interrupt. INTISRIVEC register is provided for diagnostic purposes only. It is not necessary for the ISR to read INTISRIVEC. In case of an NMI, ISR must read the INTNMIIVEC register to clear the NMI.

If the request is of the INTISR class, the corresponding vector table address is generated and presented to the CPU for vectoring. If the request is an INTNMI class interrupt, the address is fixed in the CPU to the third entry of the vector table.

All the interrupt registers are updated when a new highest interrupt line becomes active.

8.3 M3VIM *Operation*

8.3.1 *Vector Table*

Unlike other M3VIM modules in the TMS470M Series Platform architecture, the M3VIM does not have a separate RAM containing the vector table. The vector table used with the M3VIM is stored in the system memory of the Cortex M3.

The default location of the vector table is address 0x0000_0000. The user may change the base address of the vector table by programming the NVIC Vector Table Offset Register. The base address of the vector table may be any location in CODE space (flash or SRAM).

The format of the vector table differs from the format used by other ARM cores. Each entry is a 32 bit word aligned vector to an address in executable memory. Vectors are fetched on the M3 ICODE bus.

At reset, the user is responsible for providing a minimum of four entries in the M3's vector table - initial stack pointer, RESET handler vector, INTNMI handler vector, and Hard Fault handler vector. This minimizes the amount of vector space required in the boot flash memory. If these four entries are not provided, behavior of the M3 will be unpredictable.

The M3 vector table holds vectors for both tightly coupled system exceptions as well as external interrupts. The system exceptions are controlled by the M3 NVIC peripheral. The external interrupts are controlled by the M3VIM module.

Priority from NVIC system exceptions may be set from 0 to 7. Lower numbers indicate higher priority. If two exceptions share the same NVIC programmed priority a second level of arbitration will be done based on the position of the interrupt in the vector table. As with programmable priority, lower numbers indicate higher priority.

Priority from M3VIM external interrupts is determined by M3VIM channel number. Lower numbers indicate higher priority.

The figure below shows the vector table contents.

Table 8-1. Vector Table

Vector Table Address Offset	Vector Position/ Interrupt Number	Priority	Entry Name	Description
0x0	0x0	N/A	Stack Table Pointer	Address of the default stack on reset
0x4	0x1	-3	Reset	Power-on or warm reset
0x8	0x2	-2	NMI	Non Maskable Interrupt
0xC	0x3	-1	Hard Fault	Repeated or unhandlable fault
0x10	0x4	NVIC	Memory Management	MPU related fault
0x14	0x5	NVIC	Bus Fault	Memory address/data related fault
0x18	0x6	NVIC	Usage Fault	Undef. instruction or similar fault
0x1C	0x7	NVIC	Reserved	Reserved
0x20	0x8	NVIC	Reserved	Reserved
0x24	0x9	NVIC	Reserved	Reserved
0x28	0xA	NVIC	Reserved	Reserved
0x2C	0xB	NVIC	SVC Call	System call with SVC instruction
0x30	0xC	NVIC	Debug Monitor	Non-halting debug monitor
0x34	0xD	NVIC	Reserved	Reserved
0x38	0xE	NVIC	Pend SV	Software pendable service request
0x3C	0xF	NVIC	SYSTICK	System tick timer interrupt
0x40	0x10	M3VIM	External Int 0	M3VIM channel 0
0x44	0x11	M3VIM	External Int 1	M3VIM channel 1
0x48	0x12	M3VIM	External Int 2	M3VIM channel 2
...	...	...	...	...
0x23C	0x8F	M3VIM	External Int 127	M3VIM channel 127 - end of vector table in autovector mode

8.3.2 M3VIM *response to interrupt request*

8.3.2.1 *Hardware auto-vector interrupt (INTISR)*

When the M3VIM has identified an INT request as being of INTISR type, the M3VIM initiates a handshake protocol with the NVIC. In this protocol, the M3VIM initiates an autovector request. The NVIC responds requesting the address offset of the vector in the vector table. The M3VIM provides this data and the CPU vectors to the ISR.

Once the address is read, the CPU fetches the instructions from this location and begins the execution of the ISR corresponding to the highest pending INTISR request. The CPU will enter the handler state upon processing the interrupt. Typical latency is in the range of 12 CPU cycles but this will vary depending on the speed of the memory subsystem and whether the currently executing instruction must execute to completion.

The interrupt request arrives asynchronously to the M3VIM. The interrupt is then synchronized with VBUSP_CLK.

8.3.2.2 *Hardware Non-Masked Interrupt (INTNMI)*

The M3VIM handles the processing of an INTNMI identically to that of a hardware auto-vectored interrupt with one important difference. The M3VIM does not provide a vector address on generation of an INTNMI. The INTNMI is hard coded to the third entry of the vector table. Because the index of INTNMI in the vector table is fixed, there is no need for the M3VIM to NVIC handshake used with auto-vectoring interrupts.

The INTNMI has a very high priority in the system. The M3 typically responds by fetching the INTNMI ISR within three clock cycles of INTNMI input to the NVIC.

Note that only one INTNMI may be processed by the Cortex M3 NVIC at a time. In other words, an INTNMI handler will not be interrupted if a second INTNMI request is received. Because of this, the M3VIM shall drive the INTNMI request with a high level when active rather than with a pulse.

Note:
Once the M3VIM has started to drive VECTADDR, it may only change the value when ETMINTSTAT = 0x4 (vectoring state) AND ETMINTNUM is between 0x10 and 0x17. The second condition may be simplified to a simple check of ETMINTNUM[4]=1 if it is guaranteed that the design uses only autovectored interrupts connected to the M3 NVIC.

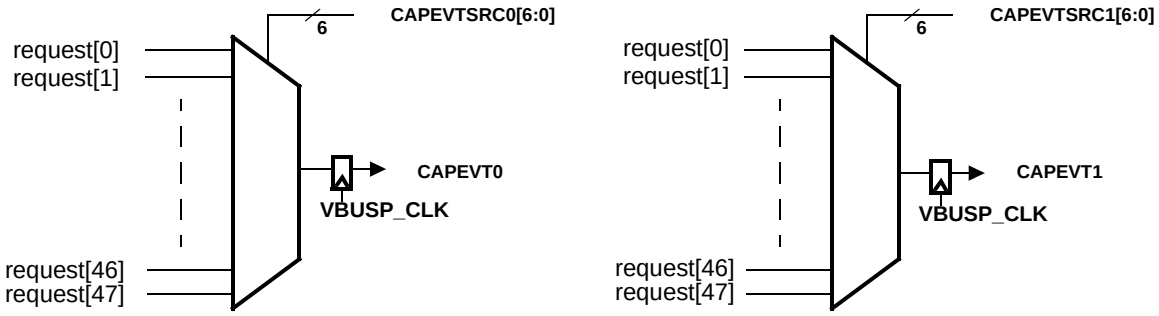
8.3.3 .*Emulation*

The M3VIM does continue to operate during debug mode; all registers are updated continuously. In debug mode, registers can be updated by the debugger independently of the CPU.

8.4 Capture event sources

The M3VIM is able to select any of the interrupt request sources in order to generate capture events for the Real Time Interrupt (RTI) module. Two registers are available to select, for each capture event source, the interrupt request line after synchronization.

Figure 8-9. Capture event sources block diagram



8.5 Programmer's Model Notes

The programmer's model for Cortex M3 NVIC autovectoring differs slightly from the programmer's model of other ARM cores. The following section highlights the major differences.

Note:
NVIC is part of the Cortex-M3. The NVIC register map starts at address 0xE000E000 and register descriptions can be found in Cortex-M3 Technical Reference Manual supplied by ARM.

8.5.1 Required Software Initialization

The startup code must perform two tasks for the autovectored interrupts to properly function.

1. The startup code must set specific priorities for the 8 implemented INTISR inputs to the NVIC. This is done to enforce nesting. Variation of priority configuration from what M3VIM expects may result in unpredictable system level results. Each interrupt has its own 8 bit priority field. The number of priority bits implemented is device dependent. The priority levels are the MSBs of each interrupt's priority field.

M3VIM is implemented with at least 3 bits of priority giving 8 priority levels, in order to support 8 levels of nesting. Some devices are built with additional priority levels which may be used for prioritization of internal exceptions. Following example is based on 4 bits of priority (16 priority levels). Please refer to the device datasheet for the number of priorities implemented.

For an implementation with 4 bits of priority, bits [7:4] of the priority register will be used. INTISR[7] must have highest priority, decreasing to INTISR[0] with lowest priority. These requirements are satisfied by programming the NVIC Interrupt Priority registers as follows (for implementation with 4 bits of priority):
0xE000E400 = 0x40506070
0xE000E404 = 0x00102030

When nesting is disabled, only the INTISR[0] autovectored interrupt is used to interrupt the core

2. The startup code must enable the 8 implemented INTISR inputs to the NVIC. Each INTISR signal is enabled by writing a one to its enable bit. This is done by programming the NVIC Interrupt Set-Enable registers as follows:

```
0xE000E100 = 0x000000FF
```

The following example code segment may be used to perform both initializations. Note that more optimal coding is possible; the code has been simplified for readability.

```
MOVW    R0, #0x6070
MOVT    R0, #0x4050
MOVW    R1, #0xE400
MOVT    R1, #0xE000
STR     R0, [R1]
MOVW    R0, #0x2030
MOVT    R0, #0x0010
MOVW    R1, #0xE404
MOVT    R1, #0xE000
STR     R0, [R1]
MOVW    R0, #0x00FF
MOVW    R1, #0xE100
MOVT    R1, #0xE000
STR     R0, [R1]
```

8.5.2 NVIC vs. M3VIM controls

- All system exceptions (NVIC numbers 0x0 through 0xF) remain controlled by NVIC due to tight coupling with the Cortex M3 processor core. The NVIC controls will be used for prioritization, set/clear, and status reporting.
- All external interrupts will be handled by the M3VIM module (NVIC numbers 0x10 through 0x8F, M3VIM channels 0 through 47). The M3VIM controls will be used for prioritization, set/clear, and status reporting.

Note: It is not recommended to modify the NVIC BASEPRI or PRIMASK registers during normal operation. Modifying these registers may result in loss of M3VIM - NVIC synchronization and unpredictable system behavior.

8.5.3 Autovectoring versus INTNMI Interrupts

M3VIM channels 0 and 1 are hardwired to generate an INTNMI. All other channels may be programmed to generate either an autovectored interrupt or an INTNMI. By default, all other M3VIM channels will generate an autovectored interrupt.

The ability to generate INTNMI from multiple sources is provided as a convenience. The user software is responsible to partition the interrupt handling for best response time. If too many sources are directed to the INTNMI then the interrupt response will become slower than with a standard autovectored interrupt.

8.5.4 Interrupt Clearing

The ISR is responsible for clearing the condition which has caused the interrupt. This is typically done with a register write to the interrupt source module. This is especially important in the case of a peripheral which generates level sensitive interrupts. The interrupt source must be cleared and the change propagated through the M3VIM to the NVIC before ISR exit. If this is not done, the program execution may tail-chain to the same ISR which was just exited due to a continued high level on the INTREQ line. In case of an NMI, ISR must read the INTNMIIVEC register to clear the NMI.

As an example, consider a simple interrupt service routine in which a timer capture register is read and then the interrupt source flag cleared. The timer module will typically use a clock source which is half the frequency of the M3 CPU. It will take a small number of CPU clock cycles for the change to reach the M3VIM, and then for the M3VIM to de-assert the INTISR signal to the NVIC. The exact number of cycles will vary with the clock frequencies used in the application. It may be necessary to pad the ISR with a few exiting NOPs to protect against an inadvertent tail chain reentry to the currently executing ISR.

8.5.5 Reset

The M3VIM and NVIC must remain synchronized during operation. If eithermodule is reset, unpredictable interrupt behavior may result. This extends to the M3 CPU core as well as NVIC is inside the boundary of the CPU. If a reset must be generated to the CPU without reset of M3VIM, it is critical that the programmer ensure that no interrupts are active before reset. This includes resets initiated by debugger.

To ensure there are no synchronization issues it is recommended that a warm systemreset (nRST) be used rather than a CPU reset. The debugger may issue this command under CCS by using the “ICEPICK Advanced Hardware Reset” function.

8.6 Registers

This section details the vectored interrupt manager module registers memory map as listed in Table 8-3 A detailed description of each register and its bits is also provided.

Each register begins on a word boundary. All registers are 32-bit, 16-bit and 8-bit accessible.

The base address for the control registers is 0xFFFF FE00

Table 8-2. Control Register Map

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
0x00 INTISRIVEC Page 291	Reserved															
	Reserved								INTISRIVEC [7:0]							
0x04 INTNMIIVEC Page 292	Reserved															
	Reserved								INTNMIIVEC [7:0]							
0x08 NEST_CTRL Page 293	Reserved															
	Reserved															NEST_ENABLE
0x0C NEST_STAT Page 294	Reserved															
	NEST_LEVEL								Reserved						NEST_STAT	NEST_OVRN
0x10 NMIPR0 Page 296	NMIPR0 [31:16]															
	NMIPR0 [15:0]															
0x14 NMIPR1 Page 297	Reserved															
	NMIPR1 [47:32]															
0x18 NMIPR2 Page 298	Reserved															
	Reserved															

Table 8-2. Control Register Map (Continued)

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
0x1C NMIPR3 Page 298	Reserved															
	Reserved															
0x20 INTREQ0 Page 299	INTREQ [31:16]															
	INTREQ [15:0]															
0x24 INTREQ1 Page 299	Reserved															
	INTREQ [47:32]															
0x28 INTREQ2 Page 299	Reserved															
	Reserved															
0x2C INTREQ3 Page 299	Reserved															
	Reserved															
0x30 REQMASKSET0 Page 302	REQMASKSET [31:16]															
	REQMASKSET [15:0]															
0x34 REQMASKSET1 Page 302	Reserved															
	REQMASKSET [47:32]															
0x38	Reserved															
	Reserved															

Table 8-2. Control Register Map (Continued)

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
0x3C	Reserved															
	Reserved															
0x40 REQMASKCLR0 Page 305	REQMASKCLR [31:16]															
	REQMASKCLR [15:0]															
0x44 REQMASKCLR1 Page 305	Reserved															
	REQMASKCLR [47:32]															
0x48	Reserved															
	Reserved															
0x4C	Reserved															
	Reserved															
0x50 WAKEMASKSET0 Page 308	WAKEMASKSET [31:16]															
	WAKEMASKSET [15:0]															
0x54 WAKEMASKSET1 Page 308	Reserved															
	WAKEMASKSET [47:32]															
0x58	Reserved															
	Reserved															

Table 8-2. Control Register Map (Continued)

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
0x5C	Reserved															
	Reserved															
0x60 WAKEMASKCLR0 Page 311	WAKEMASKCLR [31:16]															
	WAKEMASKCLR [15:0]															
0x64 WAKEMASKCLR1 Page 311	Reserved															
	WAKEMASKCLR [47:32]															
0x68	Reserved															
	Reserved															
0x6C	Reserved															
	Reserved															
0x70	Reserved															
	Reserved															
0x74	Reserved															
	Reserved															
0x78 CAPEVT Page 314	Reserved															CAPEVTSRC1 [6:0]
	Reserved															CAPEVTSRC0 [6:0]

Table 8-2. Control Register Map (Continued)

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
0x7C	Reserved															
	Reserved															
0x80 CHANCTRL0 Page 315	Reserved			CHANMAP0 [6:0]					Reserved			CHANMAP1 [6:0]				
	Reserved			CHANMAP2 [6:0]					Reserved			CHANMAP3 [6:0]				
0x84 CHANCTRL1 Page 315	Reserved			CHANMAP4 [6:0]					Reserved			CHANMAP5 [6:0]				
	Reserved			CHANMAP6 [6:0]					Reserved			CHANMAP7 [6:0]				
0x88 CHANCTRL2 Page 315	Reserved			CHANMAP8 [6:0]					Reserved			CHANMAP9 [6:0]				
	Reserved			CHANMAP10 [6:0]					Reserved			CHANMAP11 [6:0]				
0x8C CHANCTRL3 Page 315	Reserved			CHANMAP12 [6:0]					Reserved			CHANMAP13 [6:0]				
	Reserved			CHANMAP14 [6:0]					Reserved			CHANMAP15 [6:0]				
0x90 CHANCTRL4 Page 315	Reserved			CHANMAP16 [5:0]					Reserved			CHANMAP17 [6:0]				
	Reserved			CHANMAP18 [6:0]					Reserved			CHANMAP19 [6:0]				
0x94 CHANCTRL5 Page 315	Reserved			CHANMAP20 [6:0]					Reserved			CHANMAP21 [6:0]				
	Reserved			CHANMAP22 [6:0]					Reserved			CHANMAP23 [6:0]				
0x98 CHANCTRL6 Page 315	Reserved			CHANMAP24 [6:0]					Reserved			CHANMAP25 [6:0]				
	Reserved			CHANMAP26 [6:0]					Reserved			CHANMAP27 [6:0]				

Table 8-2. Control Register Map (Continued)

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
0x9C CHANCTRL7 Page 315	Reserved	CHANMAP28 [6:0]							Reserved	CHANMAP29 [6:0]						
	Reserved	CHANMAP30 [6:0]							Reserved	CHANMAP31 [6:0]						
0xA0 CHANCTRL8 Page 315	Reserved	CHANMAP32 [6:0]							Reserved	CHANMAP33 [6:0]						
	Reserved	CHANMAP34 [6:0]							Reserved	CHANMAP35 [6:0]						
0xA4 CHANCTRL9 Page 315	Reserved	CHANMAP36 [6:0]							Reserved	CHANMAP37 [6:0]						
	Reserved	CHANMAP38 [6:0]							Reserved	CHANMAP39 [6:0]						
0xA8 CHANCTRL10 Page 315	Reserved	CHANMAP40 [6:0]							Reserved	CHANMAP41 [6:0]						
	Reserved	CHANMAP42 [6:0]							Reserved	CHANMAP43 [6:0]						
0xAC CHANCTRL11 Page 315	Reserved	CHANMAP44 [6:0]							Reserved	CHANMAP45 [6:0]						
	Reserved	CHANMAP46 [6:0]							Reserved	CHANMAP47 [6:0]						
0xB0 to 0xFFFF FEFC	Reserved															
	Reserved															

8.6.1 M3VIM *Channel Offset Registers*

The M3VIM offset register provides the user with a value representing the numerical channel number of the pending interrupt with the highest precedence. INTISRIVEC holds the channel number of the highest priority active INTISR interrupt. INTISRIVEC will hold the channel number of the only active INTISR interrupt if nesting is disabled. INTNMIIVEC holds the channel number of the active INTNMI interrupt. This value must be read in order to clear the INTNMI source from the M3VIM. Table 8-3 describes the decode of this value. Reading any of these registers will not clear the interrupt flag or the offset for an INTISR class interrupt.

Table 8-3. Interrupt Dispatch Table

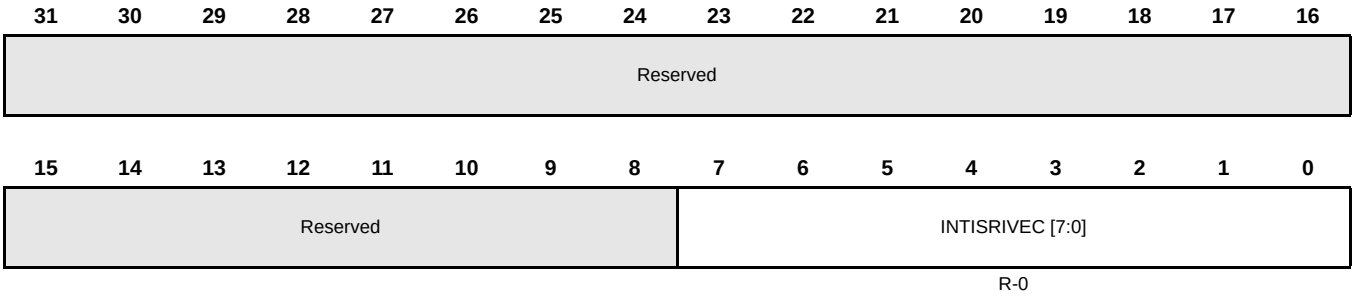
IVEC Index	Pending Interrupt
0x00	No interrupt
0x01	Channel 0
:	:
0x30	Channel 47

The M3VIM offset registers are read only. They are updated continuously by the M3VIM. Once the interrupt is serviced, the offset vectors show the value for the next highest pending interrupt or if no interrupt is pending, 0x00.

8.6.1.1 INTISR Index Offset Vector Register (INTISRIVEC)

The INTISR offset register provides the user with the channel number of the highest priority active INTISR interrupt. This register is described in Figure 8-10 and Table 8-4. The offset address is 0xFFFF_FE00.

Figure 8-10. INTISR Index Offset Vector Register (INTISRIVEC)



R = Read only; -*n* = Value after reset

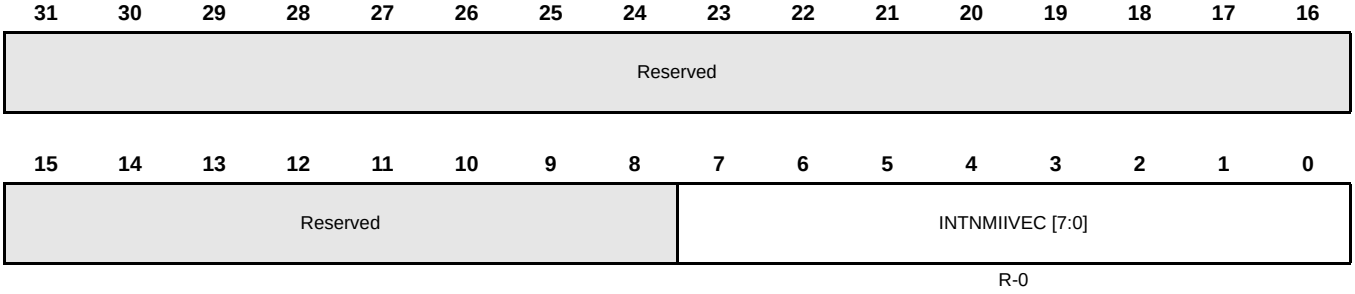
Table 8-4. INTISR Index Offset Vector Register (INTISRIVEC)

Bit	Name	Value	Description
31–8	Reserved		Reads return zero and writes have no effect.
7-0	INTISRIVEC	0–30h	Index Vector Read: Represents the channel number of the active INTISR interrupt with the highest precedence, as shown in Table 8-3 When no interrupts are pending, the least significant byte of INTISRIVEC is 0x00. Write: Writes have no effect.

8.6.1.2 NMI Index Offset Vector Register (INTNMIIVEC)

The INTNMIIVEC register provides the user with the channel number of the highest priority pending NMI interrupt. This value must be read in order to clear the INTNMI source from the M3VIM. This register is described in Figure 8-11 and Table 8-5. The offset address is 0xFFFF_FE04.

Figure 8-11. NMI Index Offset Vector Register (INTNMIIVEC)



R = Read only; *-n* = Value after reset

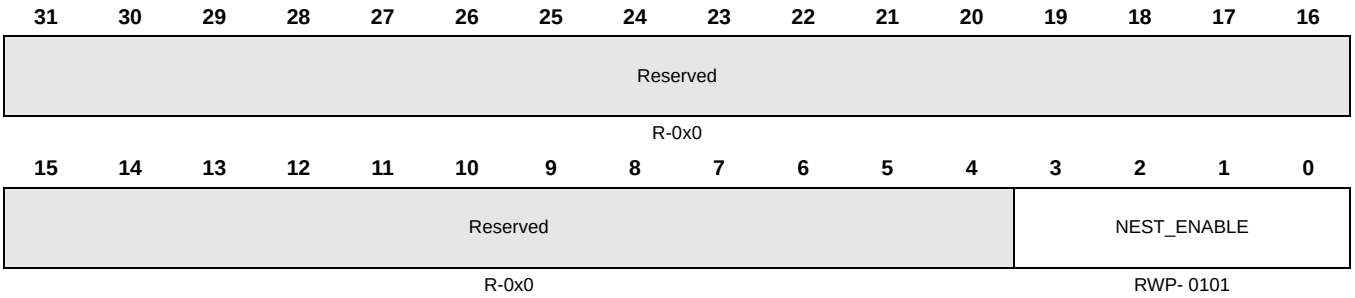
Table 8-5. NMI Index Offset Vector Register (INTISRIVEC)

Bit	Name	Value	Description
31–8	Reserved		Reads return zero and writes have no effect.
7-0	INTNMIIVEC	0–30h	Index Vector Read: Represents the channel number of the NMI pending interrupt with the highest precedence, as shown in Table 8-3 When no interrupts are pending, the least significant byte of INTNMIIVEC is 0x00. Write: Writes have no effect.

8.6.2 NESTCTRL Register

This register is described in Figure 8-12 and Table 8-6. The offset address is 0xFFFF_FE08.

Figure 8-12. NESTCTRL Register



R = Read, W = Write, WP = Write from privilege mode only, S = Set, *-n* = Value after reset (see table)

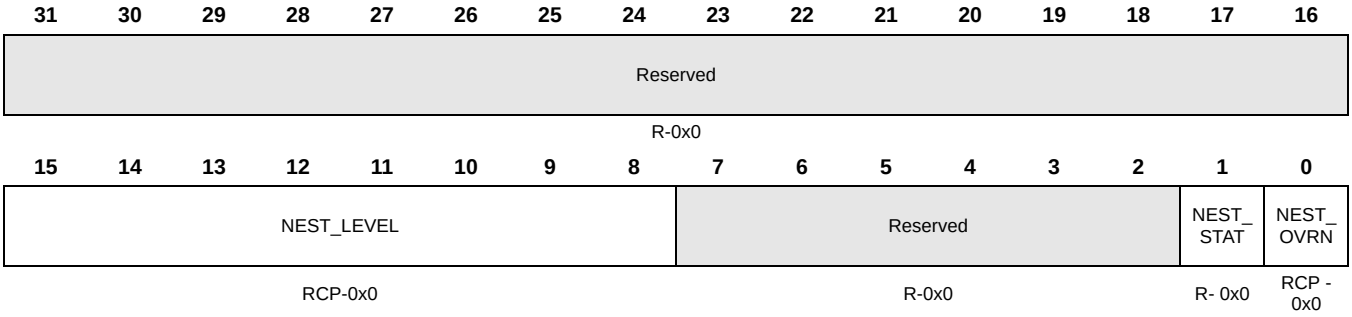
Table 8-6. NESTCTRL Register Field Descriptions

Bit	Name	Value	Description
31–4	Reserved		Reads return zero and writes have no effect.
3-0	NEST_ENABLE		Nest Enable A four bit key must be written to enable/disable nesting. **Privileged Mode Write:**
		0101	Disable Nesting
		1010	Enable Nesting
			User Mode Write: No effect

8.6.3 NESTSTAT Register

This register is described in Figure 8-13 and Table 8-7. The offset address is 0xFFFF_FE0C.

Figure 8-13. NESTSTAT Register



R = Read, W = Write, WP = Write from privilege mode only, CP = Clear from privilege mode only, S = Set, *-n* = Value after reset (see table)

Table 8-7. NESTSTAT Register Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Reads return zero and writes have no effect.
15-8	NEST_LEVEL		Nest Level These bits show the highest level of nesting achieved. **User and Privileged Mode Read:**
		0	No nesting
		1	1st nest level reached
		2	2nd nest level reached
		...	...
		128	8th nest level reached
			Privileged Mode Write:
		0	No effect
		1	Clear Bits
			User Mode Write:
			No effect
7-2	Reserved		Reads return zero and writes have no effect.
1	NEST_STAT		Nest Status This bit shows if nesting is enabled. Writes have no effect.
		0	Nesting disabled
		1	Nesting enabled

Table 8-7. NESTSTAT Register Field Descriptions (Continued)

Bit	Name	Value	Description
0	NEST_OVRN		Nesting Level Over-Run This bit shows if all nesting levels are used and another interrupt is pending. **User and privilege mode read:**
		0	Nesting not full
		1	Nesting Full
			Privilege mode write:
		0	No effect
		1	Clear the NEST_OVRN bit
			User Mode Write:
			No effect

8.6.4 INTNMI/INTISR Program Control Registers (NMIPRx)

INTNMI/INTISR program control register (NMIPR), determines whether a given channel would generate an interrupt request of type INTNMI or INTISR. These registers are described in Figure 8-14 through Figure 8-17 and Table 8-8 through Table 8-11. The offset address for NMIPR0 is 0xFFFF_FE10.

Figure 8-14. *NMI0 Program Control Register (NMIPR0)*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NMIPR0 [31:16]															
RWP-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NMIPR0 [15:0]															
RWP-0															

Table 8-8. NMI0 Program Control Register (NMIPR0) Field Descriptions

Bit	Name	Value	Description
31-0	NMIPR0 [31:0]		INTNMI/INTISR program control bits These bits determine whether an channel would generate an interrupt request of type INTNMI or INTISR. Each bit corresponds to one request channel between 0 and 31. NUMIPR0 [0] = Request Channel 0 ... NUMIPR0 [31] = Request Channel 31 **User and privilege mode read:**
		0	Interrupt request is of INTISR type
		1	Interrupt request is of INTNMI type
			Privilege mode write:
		0	Interrupt request is of INTISR type
		1	Interrupt request is of INTNMI type
			Note: NMIPR0[1:0] are reserved channels which may only generate an NMI type interrupt. As such they are read-only. Reads will return a value of ‘1’.

The offset address for NMIPR1 is 0xFFFF_FE14.

Figure 8-15. *NMI1 Program Control Register (NMIPR1)*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NMIPR1 [47:32]															
RWP-0															

Table 8-9. NMI1 Program Control Register (NMIPR1) Field Descriptions

Bit	Name	Value	Description
31-16	Reserved		Reads return zero and writes have no effect.
15-0	NMIPR1 [47:32]		INTNMI/INTISR program control bits These bits determine whether an channel would generate an interrupt request of type INTNMI or INTISR. Each bit corresponds to one request channel between 32 and 63. NUMIPR1 [0] = Request Channel 32 ... NUMIPR1 [31] = Request Channel 63 **User and privilege mode read:**
		0	Interrupt request is of INTISR type
		1	Interrupt request is of INTNMI type
			Privilege mode write:
		0	Interrupt request is of INTISR type
		1	Interrupt request is of INTNMI type
			Note: NMIPR0[1:0] are reserved channels which may only generate an NMI type interrupt. As such they are read-only. Reads will return a value of ‘1’.

The offset address for NMIPR2 is 0xFFFF_FE18.

Figure 8-16. *NMI2 Program Control Register (NMIPR2)*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserve															
R-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved															
R-0															

Table 8-10. NMI2 Program Control Register (NMIPR2) Field Descriptions

Bit	Name	Value	Description
31-0	NMIPR2 [95:64]		Reserved

The offset address for NMIPR3 is 0xFFFF_FE1C.

Figure 8-17. *NMI3 Program Control Register (NMIPR3)*

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

R = Read in all modes, WP = Writable in privilege mode only; *-n* = Value after reset

Table 8-11. NMI3 Program Control Register (NMIPR3) Field Descriptions

Bit	Name	Value	Description
31-0	NMIPR3 [127:96]		Reserved

8.6.5 Pending Interrupt Read Location Registers (INTREQx)

The pending interrupt registers gives the pending interrupt requests. The registers are updated every vbus clock cycle. These registers are described in Figure 8-18 through Figure 8-21 and Table 8-12 through Table 8-15. The offset address for INTREQ0 is 0xFFFF_FE20.

Figure 8-18. Pending Interrupt Read Location Register 0 (INTREQ0)

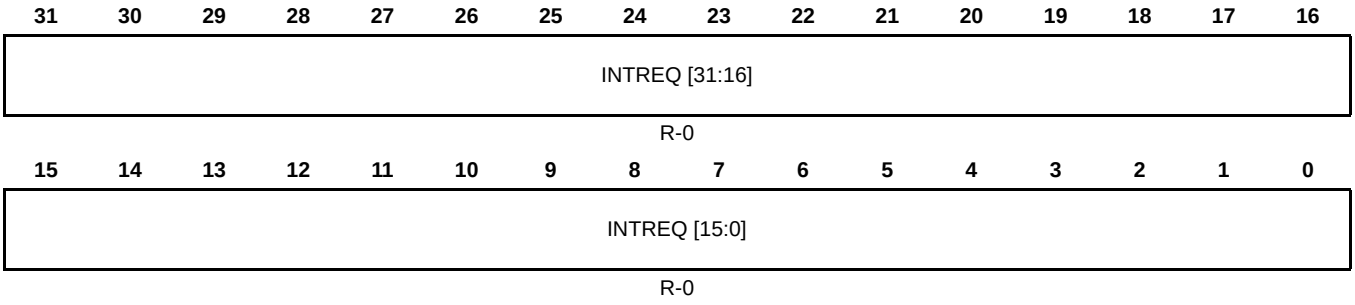


Table 8-12. Pending Interrupt Read Location Register 0 (INTREQ0) Field Descriptions

Bit	Name	Value	Description
31-0	INTREQ0 [31:0]		Pending Interrupt bits These bits determine whether an interrupt request is pending for the request channel between 0 and 31. The interrupt mask register does not affect the value of the interrupt pending bit. INTREQ0[0] = Request Channel 0 ... INTREQ0[31] = Request Channel 31 **Reads:**
		0	No interrupt event has occurred
		1	Interrupt is pending
			Writes:
			Writes have no effect

The offset address for INTREQ1 is 0xFFFF_FE24.

Figure 8-19. Pending Interrupt Read Location Register 1 (INTREQ1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
INTREQ [47:32]															
R-0															

Table 8-13. Pending Interrupt Read Location Register 1 (INTREQ1) Field Descriptions

Bit	Name	Value	Description
31-16	Reserved		Reads return zero and writes have no effect.
15-0	INTREQ1 [47:32]		Pending Interrupt bits These bits determine whether an interrupt request is pending for the request channel between 0 and 31. The interrupt mask register does not affect the value of the interrupt pending bit. INTREQ1[0] = Request Channel 0 ... INTREQ1[31] = Request Channel 31 **Reads:**
		0	No interrupt event has occurred
		1	Interrupt is pending
			Writes:
			Writes have no effect

The offset address for INTREQ2 is 0xFFFF_FE28.

Figure 8-20. Pending Interrupt Read Location Register 2 (INTREQ2)

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

Table 8-14. Pending Interrupt Read Location Register 2 (INTREQ2) Field Descriptions

Bit	Name	Value	Description
31-0	INTREQ2 [95:64]		Reserved

The offset address for INTREQ3 is 0xFFFF_FE2C.

Figure 8-21. Pending Interrupt Read Location Register 3 (INTREQ3)

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

R = Read in all modes, *-n* = Value after reset

Table 8-15. Pending Interrupt Read Location Register 3 (INTREQ3) Field Descriptions

Bit	Name	Value	Description
31-0	INTREQ3 [127:96]		Reserved

8.6.6 Interrupt Mask Set Registers (REQMASKSETx)

The interrupt register mask selectively enables individual request channels.These registers are described in Figure 8-22 through Figure 8-25 and Table 8-16 through Table 8-19. The offset address for REQMASKSET0 is 0xFFFF_FE30.

Figure 8-22. Interrupt Mask Set Register 0 (REQMASKSET0)

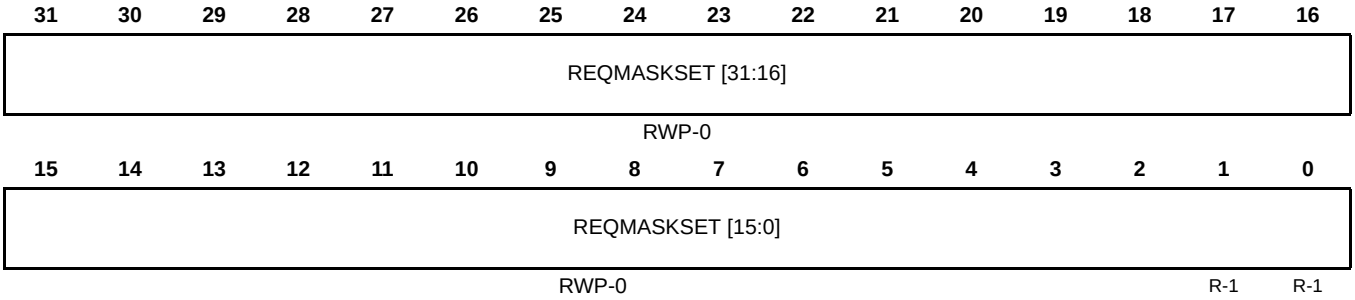


Table 8-16. Interrupt Mask Set Register 0 (REQMASKSET0) Field Descriptions

Bit	Name	Value	Description
31-0	REQMASKSET0 [31:0]		Request Mask Set bits This vector determines whether the interrupt channel is enabled. **Note : Channel 0 and 1 are configured as NMI. *Hence REQMASKSET0 [0] and REQMASKSET0 [1] bits are Read Only and always read as 1.*** REQMASKSET0 [0] = Request Channel 0 ... REQMASKSET0 [31] = Request Channel 31 **User and privilege mode read:**
		0	Interrupt request channel is disabled
		1	Interrupt request channel is enabled
			Privilege mode write:
		0	no effect
		1	Interrupt request channel is enabled

The offset address for REQMASKSET1 is 0xFFFF_FE34.

Figure 8-23. Interrupt Mask Set Register 1 (REQMASKSET1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
REQMASKSET [47:32]															
RWP-0															

Table 8-17. Interrupt Mask Set Register 1 (REQMASKSET1) Field Descriptions

Bit	Name	Value	Description
31-16	Reserved		Reads return zero and writes have no effect.
15-0	REQMASKSET1 [47:32]		Request Mask Set bits This vector determines whether the interrupt channel is enabled. REQMASKSET1[0] = Request Channel 0 ... REQMASKSET1 [31] = Request Channel 31 **User and privilege mode read:**
		0	Interrupt request channel is disabled
		1	Interrupt request channel is enabled
			Privilege mode write:
		0	no effect
		1	Interrupt request channel is enabled

The offset address for REQMASKSET2 is 0xFFFF_FE38.

Figure 8-24. Interrupt Mask Set Register 2 (REQMASKSET2)

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

Table 8-18. Interrupt Mask Set Register 2 (REQMASKSET2) Field Descriptions

Bit	Name	Value	Description
31-0	REQMASKSET2 [95-64]		Reserved

The offset address for REQMASKSET3 is 0xFFFF_FE3C.

Figure 8-25. Interrupt Mask Set Register 3 (REQMASKSET3)

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

R = Read in all modes, WP = Writable in privilege mode only; *-n* = Value after reset

Table 8-19. Interrupt Mask Set Register 3 (REQMASKSET3) Field Descriptions

Bit	Name	Value	Description
31-0	REQMASKSET3 [127-96]		Reserved

8.6.7 Interrupt Mask Clear Registers (REQMASKCLRx)

The interrupt register mask selectively disables individual request channels.These registers are described in Figure 8-26 through Figure 8-29 and Table 8-20 through Table 8-23. The offset address for REQMASKCLR0 is 0xFFFF_FE40.

Figure 8-26. Interrupt Mask Clear Register 0 (REQMASKCLR0)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REQMASKCLR [31:16]															
RWP-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
REQMASKCLR [15:0]															
RWP-0															

Table 8-20. Interrupt Mask Clear Register 0 (REQMASKCLR0) Field Descriptions

Bit	Name	Value	Description
31-0	REQMASKCLR0		Request Mask Clear bits This vector determines whether the interrupt request channel is enabled. **Note : Channel 0 and 1 are configured as NMI. *Hence REQMASKCLR0 [0] and REQMASKCLR0 [1] bits are Read Only and always read as 1.*** REQMASKCLR0 [0] = Request Channel 0 ... REQMASKCLR0 [31] = Request Channel 31 **User and privilege mode read:**
		0	Interrupt request channel is disabled
		1	Interrupt request channel is enabled
			Privilege mode write:
		0	no effect
		1	Interrupt request channel is disabled

The offset address for REQMASKCLR1 is 0xFFFF_FE44.

Figure 8-27. Interrupt Mask Clear Register 1 (REQMASKCLR1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
REQMASKCLR [47:32]															
RWP-0															

Table 8-21. Interrupt Mask Clear Register 1 (REQMASKCLR1) Field Descriptions

Bit	Name	Value	Description
31-16	REQMASKCLR1 [63-48]		Reserved
15-0	REQMASKCLR1 [47:32]		Request Mask Clear bits This vector determines whether the interrupt request channel is enabled. REQMASKCLR1 [0] = Request Channel 0 ... REQMASKCLR1 [31] = Request Channel 31 **User and privilege mode read:**
		0	Interrupt request channel is disabled
		1	Interrupt request channel is enabled
			Privilege mode write:
		0	no effect
		1	Interrupt request channel is disabled

The offset address for REQMASKCLR2 is 0xFFFF_FE48.

Figure 8-28. Interrupt Mask Clear Register 2 (REQMASKCLR2)

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

Table 8-22. Interrupt Mask Clear Register 2 (REQMASKCLR2) Field Descriptions

Bit	Name	Value	Description
31-0	REQMASKCLR2 [95-64]		Reserved

The offset address for REQMASKCLR3 is 0xFFFF_FE4C.

Figure 8-29. Interrupt Mask Clear Register 2 (REQMASKCLR2)

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

R = Read in all modes, WP = Writable in privilege mode only; *-n* = Value after reset

Table 8-23. Interrupt Mask Clear Register 3 (REQMASKCLR3) Field Descriptions

Bit	Name	Value	Description
31-0	REQMASKCLR3 [127-96]		Reserved

8.6.8 Wake-up Mask Set Registers (WAKEMASKSETx)

The wake-up mask register selectively enables individual wake-up interrupt request lines. These registers are described in Figure 8-30 through Figure 8-33 and Table 8-24 through Table 8-27. The offset address for WAKEMASKSET0 is 0xFFFF_FE50.

Figure 8-30. Wake-up Mask Set Register 0 (WAKEMASKSET0)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WAKEMASKSET [31:16]															
RWP-0xFFFF															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
WAKEMASKSET [15:0]															
RWP-0xFFFF															

Table 8-24. Wake-Up Mask Set Register 0 (WAKEMASKSET0) Field Descriptions

Bit	Name	Value	Description
31-0	WAKEMASKSET0		Wake-Up Mask Set bits This vector determines whether the wake-up interrupt line is enabled. WAKEMASKSET0[0] = Wake-up interrupt line 0 ... WAKEMASKSET1[31] = Wake-up interrupt line 31 **User and privilege mode read:**
		0	Wake-up interrupt line is disabled
		1	Wake-up interrupt line is enabled
			Privilege mode write:
		0	no effect
		1	Wake-up interrupt line is enabled

The offset address for WAKEMASKSET1 is 0xFFFF_FE54.

Figure 8-31. Wake-up Mask Set Register 1 (WAKEMASKSET1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
WAKEMASKSET [47:32]															
RWP-0xFFFF															

Table 8-25. Wake-Up Mask Set Register 1 (WAKEMASKSET1) Field Descriptions

Bit	Name	Value	Description
31-16	WAKEMASKSET1 [63-48]		Reserved
15-0	WAKEMASKSET1		Wake-Up Mask Set bits This vector determines whether the wake-up interrupt line is enabled. WAKEMASKSET1[0] = Wake-up interrupt line 0 ... WAKEMASKSET1[31] = Wake-up interrupt line 31 **User and privilege mode read:**
		0	Wake-up interrupt line is disabled
		1	Wake-up interrupt line is enabled
			Privilege mode write:
		0	no effect
		1	Wake-up interrupt line is enabled

The offset address for WAKEMASKSET2 is 0xFFFF_FE58.

Figure 8-32. Wake-up Mask Set Register 2 (WAKEMASKSET2)

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

Table 8-26. Wake-Up Mask Set Register 2 (WAKEMASKSET2) Field Descriptions

Bit	Name	Value	Description
31-0	WAKEMASKSET2 [95-64]		Reserved

The offset address for WAKEMASKSET3 is 0xFFFF_FE5C.

Figure 8-33. Wake-up Mask Set Register 3 (WAKEMASKSET3)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved WAKEMASKSET [127:112]															
R-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved WAKEMASKSET [111:96]															
R-0															

R = Read in all modes, WP = Writable in privilege mode only; *-n* = Value after reset

Table 8-27. Wake-Up Mask Set Register 3 (WAKEMASKSET3) Field Descriptions

Bit	Name	Value	Description
31-0	WAKEMASKSET3 [127-96]		Reserved

8.6.9 Wake-up Mask Clear Registers (WAKEMASKCLRx)

The wake-up mask register selectively disables individual wake-up interrupt request lines.These registers are described in Figure 8-34 through Figure 8-37 and Table 8-28 through Table 8-31. The offset address for WAKEMASKCLR0 is 0xFFFF_FE60.

Figure 8-34. Wake-up Mask Clear Register 0 (WAKEMASKCLR0)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WAKEMASKCLR [31:16]															
RWP-0xFFFF															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
WAKEMASKCLR [15:0]															
RWP-0xFFFF															

Table 8-28. Wake-Up Mask Clear Register (WAKEMASKCLR0) Field Descriptions

Bit	Name	Value	Description
63-0	WAKEMASKCLR0		Wake-Up Mask Clear bits This vector determines whether the wake-up interrupt line is enabled. WAKEMASKCLR0[0] = Wake-up interrupt line 0 ... WAKEMASKCLR0[31] = Wake-up interrupt line 31 **User and privilege mode read:**
		0	Wake-up interrupt line is disabled
		1	Wake-up interrupt line is enabled
			Privilege mode write:
		0	no effect
		1	Wake-up interrupt line is enabled

The offset address for WAKEMASKCLR1 is 0xFFFF_FE64.

Figure 8-35. Wake-up Mask Clear Register 1 (WAKEMASKCLR1)

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WAKEMASKCLR [47:32]															
RWP-0xFFFF															

Table 8-29. Wake-Up Mask Clear Register 1 (WAKEMASKCLR1) Field Descriptions

Bit	Name	Value	Description
31-16	WAKEMASKCLR1 [63-48]		Reserved
15-0	WAKEMASKCLR1 [47-32]		Wake-Up Mask Clear bits This vector determines whether the wake-up interrupt line is enabled. WAKEMASKCLR1[0] = Wake-up interrupt line 0 ... WAKEMASKCLR1[31] = Wake-up interrupt line 31 **User and privilege mode read:**
		0	Wake-up interrupt line is disabled
		1	Wake-up interrupt line is enabled
			Privilege mode write:
		0	no effect
		1	Wake-up interrupt line is enabled

The offset address for WAKEMASKCLR2 is 0xFFFF_FE68.

Figure 8-36. Wake-up Mask Clear Register 2 (WAKEMASKCLR2)

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

Table 8-30. Wake-Up Mask Clear Register 2 (WAKEMASKCLR2) Field Descriptions

Bit	Name	Value	Description
31-0	WAKEMASKCLR2 [95-64]		Reserved

The offset address for WAKEMASKCLR3 is 0xFFFF_FE6C.

Figure 8-37. Wake-up Mask Clear Register 3 (WAKEMASKCLR3)

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

R = Read in all modes, WP = Writable in privilege mode only; *-n* = Value after reset

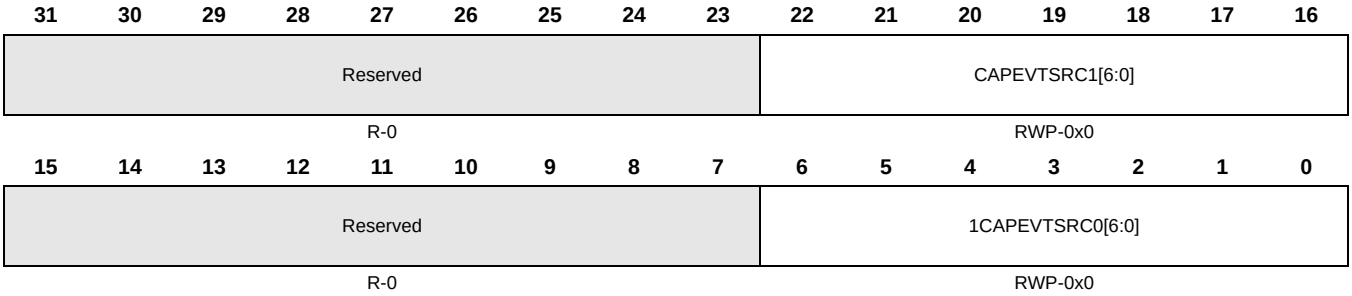
Table 8-31. Wake-Up Mask Clear Register 3 (WAKEMASKCLR3) Field Descriptions

Bit	Name	Value	Description
31-0	WAKEMASKCLR3 [127-96]		Reserved

8.6.10 Capture Event Register (CAPEVT)

This register is described in Figure 8-38 and Table 8-32. The offset address for CAPEVT is 0xFFFF_FE78.

Figure 8-38. Capture Event Register (CAPEVT)



R = Read, W = Write, WP = Write from privilege mode only, S = Set, *-n* = Value after reset (see table)

Table 8-32. Capture Event Register (CAPEVT) Field Descriptions

Bit	Name	Value	Description
31-23	Reserved		Reads return zero and writes have no effect.
22-16	CAPEVTSRC1		Capture Event Source Mapping Control 1 These bits determine which interrupt request maps to the capture event source 1 of the RTI. **User and privilege mode read:**
		0000000	Interrupt Request 0
		0000001	Interrupt Request 1
		...	...
		0101111	Interrupt Request 47
			Privilege mode write: Set the interrupt request line to be forwarded to the RTI
15-7	Reserved		Reads return zero and writes have no effect.
6-0	CAPEVTSRC0		Capture Event Source Mapping Control 0 These bits determine which interrupt request maps to the capture event source 0 of the RTI. **User and privilege mode read:**
		0000000	Interrupt Request 0
		0000001	Interrupt Request 1
		...	...
		0101111	Interrupt Request 48
			Privilege mode write: Set the interrupt request line to be forwarded to the RTI

8.6.11 M3VIM *Interrupt Control Register x (CHANCTRL x)*

The interrupt control register specifies the mapping of an interrupt request (from peripheral) to a specific interrupt channel. There are 12 interrupt control registers that control the 48 interrupt channels of the M3VIM. Each register contains the control for 4 interrupt channels. The following table shows the organization of the registers and the reset value of each.

Note that a single interrupt request can be mapped to multiple interrupt channels but not vice versa.

Table 8-33. Interrupt control registers organization

Offset Address	Mnemonic	CHANMAPx$_0$	CHANMAPx$_1$	CHANMAPx$_2$	CHANMAPx$_3$	Reset value
0x80	CHANCTRL0	CHANMAP0	CHANMAP1	CHANMAP2	CHANMAP3	0x00010203
0x84	CHANCTRL1	CHANMAP4	CHANMAP5	CHANMAP6	CHANMAP7	0x04050607
0x88	CHANCTRL2	CHANMAP8	CHANMAP9	CHANMAP10	CHANMAP11	0x08090A0B
0x8C	CHANCTRL3	CHANMAP12	CHANMAP13	CHANMAP14	CHANMAP15	0x0C0D0E0F
0x90	CHANCTRL4	CHANMAP16	CHANMAP17	CHANMAP18	CHANMAP19	0x10111213
0x94	CHANCTRL5	CHANMAP20	CHANMAP21	CHANMAP22	CHANMAP23	0x14151617
0x98	CHANCTRL6	CHANMAP24	CHANMAP25	CHANMAP26	CHANMAP27	0x18191A1B
0x9C	CHANCTRL7	CHANMAP28	CHANMAP29	CHANMAP30	CHANMAP31	0x1C1D1E1F
0xA0	CHANCTRL8	CHANMAP32	CHANMAP33	CHANMAP34	CHANMAP35	0x20212223
0xA4	CHANCTRL9	CHANMAP36	CHANMAP37	CHANMAP38	CHANMAP39	0x24252627
0xA8	CHANCTRL10	CHANMAP40	CHANMAP41	CHANMAP42	CHANMAP43	0x28292A2B
0xAC	CHANCTRL11	CHANMAP44	CHANMAP45	CHANMAP46	CHANMAP47	0x2C2D2E2F
0xB0 to 0xFC	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved

Each 4 fields of the register have been named with a generic index that refers to the detailed register organization.

8.6.11.1 Interrupt Control Register

This register is described in Figure 8-39 and Table 8-34. The offset address for this register is 0xFFFF_FE80.

Figure 8-39. Interrupt Control Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	CHANMAPx$_0$[6:0]							Reserved	CHANMAPx$_1$[6:0]						
R-0	RWP-see table							R-0	RWP-see table						
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	CHANMAPx$_2$[6:0]							Reserved	CHANMAPx$_3$[6:0]						
R-0	RWP-see table							R-0	RWP-see table						

R = Read, W = Write, WP = Write from privilege mode only, S = Set, *-n* = Value after reset (see table)

Table 8-34. Interrupt Control Register Field Descriptions

Bit	Name	Value	Description
31	Reserved		Reads return zero and writes have no effect.
30-24	CHANMAPx$_0$		Interrupt CHAN**x$_0$** Mapping Control These bits determine which interrupt request the priority channel CHAN**x$_0$** maps to **User and privilege mode read:**
		0000000	Interrupt Request 0 maps to Channel priority CHAN**x$_0$**
		0000001	Interrupt Request 1 maps to Channel priority CHAN**x$_0$**
		...	...
		0101111	Interrupt Request 47 maps to Channel priority CHAN**x$_0$**
			Privilege mode write: The default value of these bits after reset is given in the Table 8-33. Set the Channel priority CHAN**x$_0$** with the interrupt request.
23	Reserved		Reads return zero and writes have no effect.

Table 8-34. Interrupt Control Register Field Descriptions (Continued)

Bit	Name	Value	Description
22-16	CHANMAPx$_1$		Interrupt CHAN**x$_1$** Mapping Control These bits determine which interrupt request the priority channel CHAN**x$_1$** maps to **User and privilege mode read:**
		0000000	Interrupt Request 0 maps to Channel priority CHAN**x$_1$**
		0000001	Interrupt Request 1 maps to Channel priority CHAN**x$_1$**
		...	...
		0101111	Interrupt Request 47 maps to Channel priority CHAN**x$_1$**
			Privilege mode write: The default value of these bits after reset is given in the Table 8-33. Set the Channel priority CHAN**x$_1$** with the interrupt request.
15	Reserved		Reads return zero and writes have no effect.
14-8	CHANMAPx$_2$		Interrupt CHAN**x$_2$** Mapping Control These bits determine which interrupt request the priority channel CHAN**x$_2$** maps to **User and privilege mode read:**
		0000000	Interrupt Request 0 maps to Channel priority CHAN**x$_2$**
		0000001	Interrupt Request 1 maps to Channel priority CHAN**x$_2$**
		...	...
		0101111	Interrupt Request 47 maps to Channel priority CHAN**x$_2$**
			Privilege mode write: The default value of these bits after reset is given in the Table 8-33. Set the Channel priority CHAN**x$_2$** with the interrupt request.
7	Reserved		Reads return zero and writes have no effect.

Table 8-34. Interrupt Control Register Field Descriptions (Continued)

Bit	Name	Value	Description
6-0	CHANMAPx_3		Interrupt CHAN$\mathbf{x_3}$ Mapping Control These bits determine which interrupt request the priority channel CHAN$\mathbf{x_3}$ maps to **User and privilege mode read:**
		0000000	Interrupt Request 0 maps to Channel priority CHAN$\mathbf{x_3}$
		0000001	Interrupt Request 1 maps to Channel priority CHAN$\mathbf{x_3}$
		...	...
		0101111	Interrupt Request 47 maps to Channel priority CHAN$\mathbf{x_3}$
			Privilege mode write: The default value of these bits after reset is given in the Table 8-33. Set the Channel priority CHAN$\mathbf{x_3}$ with the interrupt request.

Cyclic Redundancy Check Controller (CRC) Module

Topic		Page
9.1	Overview	320
9.2	TMS470M Series CRC Features	321
9.3	Module Operation	322
9.4	CRC Control Registers	325

9.1 Overview

The CRC controller performs cyclic redundancy checks (CRC) to verify the integrity of the memory system. A signature representing the contents of the memory is obtained when the contents of the memory are read into the CRC controller. The CRC controller calculates the signature for a set of data and then compares the calculated signature value against a predetermined good signature value. The TMS470M Series CRC controller provides one channel to perform CRC calculations on multiple memories in parallel, and it can be used on any memory system. Channel 1 can also be put into data trace mode. In data trace mode, the CRC controller compresses all data being read through the CPU read data bus.

9.2 TMS470M Series CRC Features

The TMS470M Series CRC controller offers:

- One channel to perform background signature verification on any memory sub-system
- Data compression on 8, 16, 32, and 64-bit data size
- Maximum-length parallel signature analysis (PSA) register constructed based on 64-bit primitive polynomials
- Programmable 20-bit pattern counter per channel to count the number of data patterns for compression
- Interface supported: AHB
- AHB slave interface
- Data trace capability on the CPU data bus

9.3 Module Operation

The following sections describe how the module operates.

9.3.1 General Operation

A CRC controller has one channel. This channel has a memory-mapped PSA signature register.

A memory can be organized into multiple sectors with each sector consisting of multiple data patterns. A data pattern can be of 8, 16, 32, or 64-bit data. The CRC controller performs the signature calculation and compares the signature to a predetermined value. The PSA signature register compresses an incoming data pattern into a signature when it is written.

When one sector of data patterns is written into the PSA signature register, a final signature corresponding to that sector is obtained. The calculated signature and the predetermined signature are then compared to each other for signature verification.

The CRC controller also provides data trace capability. Channel 1 can perform data trace on the CPU data bus. During data trace, channel 1 monitors any data being read on the CPU data bus and compresses it. When data trace is enabled for channel 1, all circuits related to interrupt generation and counters are disabled.

9.3.2 Modes

The CRC controller can operate in data trace mode.

9.3.2.1 Data trace mode

CRC channel 1 can be used to snoop the CPU read data bus. Data read on the CPU bus is converted into a 64-bit data value where the LSB or the MSB bits of the read data bus are padded with zeros. The PSA signature register concatenates the data value with zeros to create a 64-bit word based on the address of the read transaction. For example, the data value 0x12345678 is stored at addresses 0x0 and 0x4. If the CPU reads a word from address 0x0, the value 0x0000000012345678 will be used for the CRC signature generation. If the CPU reads a word from address 0x4, the value 0x1234567800000000 will be used for the signature generation. Each data pattern read by the CPU on its data bus is compressed in the PSA signature register. To enable the data trace on the read bus, 0x0 should be written in CH1_MODE bits. The CH1TRACEEN bit should be set to 0x1. Before the data compression begins, a seed value should be placed in the PSA signature register. To change the seed value, the CH1_MODE bits should be set to 0x00 and data trace bit should be disabled. During data trace mode, all interrupts logic are inactive.

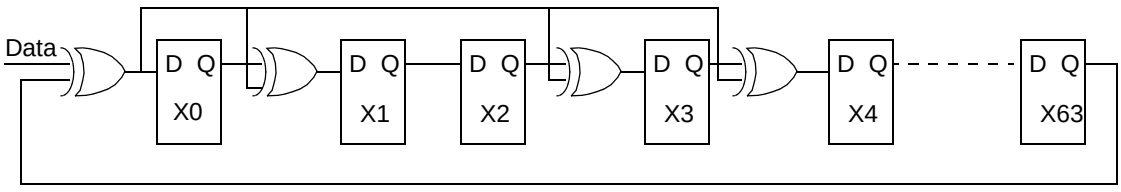
9.3.3 Register Definitions

9.3.3.1 PSA Signature Register

The 64-bit PSA signature register is based on the primitive polynomial shown in Equation 1 to produce the maximum length linear feedback shift register (LFSR) (more details of the 64-bit primitive polynomial can be found in Stahnke 1973).

$$f(x) = x^{64} + x^4 + x^3 + x + 1 \quad \text{(EQ 1)}$$

Figure 9-1. 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. It produces 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* shifts.

The parallel CRC calculation based on the above polynomial is illustrated in the (HDL) code below.

```
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;
```

1 The inner loop calculates the next value of each shift register bit after one cycle.
2 The outer loop simulates 64 cycles of shifting. The equation for each shift register bit is thus built before it is compressed into the shift register.
3 The MSB of the data is shifted in first.

PSA_REGL1[31:0] and PSA_REGH1[63:32] contain the calculated 64-bit PSA signature value for channel 1. The PSA signature register can be read from and written to. When a value is written, it can either compress the data or just capture the data depending on the state of CHx_MODE bits in the CRC_CTRL2 register. If CHx_MODE is set to data capture, a seed value can be planted in the PSA signature register without compression. The CRC channel can be planted with a different seed value before compression starts. Modes other than data capture mode will cause the data to be compressed by the PSA signature register when it is written. When the PSA signature register is read, it gives the calculated signature value.

The CRC controller supports doubleword, word, half word, and byte accesses to the PSA signature register. During a non-doubleword write access, all unwritten byte lanes are padded with zeros before compression.

There is a software reset for the PSA signature register channel. When set, the PSA signature register is reset to all zeros by writing to the CRC_CTRL0 register.

The PSA signature register is reset to zero under any of the following conditions:

- The system resets.
- A PSA software reset occurs.
- One sector of data patterns is compressed.

9.3.3.2 PSA Sector Signature Register

After one sector of data is compressed, the final resulting signature calculated by the PSA signature register is transferred to the PSA sector signature registers PSA_SECSIGHx and PSA_SECSIGLx. The PSA sector signature register is a read-only register.

9.3.3.3 Raw Data Register

The raw or uncompressed data written to the PSA signature register is also saved in the raw data register. The raw data registers, RAW_DATAREGHx and RAW_DATAREGLx, are read only.

9.3.4 Power-Down Mode

The CRC module can be put into power-down mode by setting the power-down control bit, PWDN, in the CRC_CTRL1 register. The module wakes up when the PWDN bit is cleared. See the TMS470M Series device-specific data sheet for the actual implementation of your device.

9.3.5 Emulation

During emulation mode, a read access from any register returns the register contents to the bus and will not trigger or mask any event as it would in functional mode. This prevents the debugger from disturbing the runtime environment. For example, when the debugger reads the interrupt offset register during a screen refresh, the corresponding interrupt status flag will not be cleared. Time-out counters are stopped in emulation mode. When channel 1 is configured to data trace mode, the PSA signature register does not compress any data read on the CPU data bus in emulation mode. No CPU bus error will be generated if reading from unimplemented locations in emulation mode.

9.4 CRC Control Registers

All registers are on word boundaries. 64, 32, 16, and 8 bit write accesses are supported to all registers. Figure 9-2 shows the summary of the registers. The base address for the control registers is 0xFE00 0000.

Figure 9-2. CRC Register Summary

Offset Address Register(1)	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
0x00 CRC_CTRL0 Page 327	Reserved															
	Reserved															CH1_PSA_SWREST
0x08 CRC_CTRL1 Page 328	Reserved															
	Reserved															PWDN
0x010 CRC_CTRL2 Page 329	Reserved															
	Reserved											CH1_TRACEEN	Reserved		CH1_MODE	
0x060 PSA_SIGREGL1 Page 331	PSASIG1(31–16)															
	PSASIG1(15–0)															
0x064 PSA_SIGREGH1 Page 332	PSASIG1(63–48)															
	PSASIG1(47–32)															
0x070 PSA_SECSIGREGL1 Page 333	PSASECSIG1(31–16)															
	PSASECSIG1(15–0)															
0x074 PSA_SECSIGREGH1 Page 333	PSASECSIG1(63–48)															
	PSASECSIG1(47–32)															
0x078 RAW_DATAREGL1 Page 335	RAW_DATA1(31–16)															
	RAW_DATA1(15–0)															

1 The physical address (base + offset) of these registers is device specific. See the device-specific data sheet to verify the register base address.

Figure 9-2. CRC Register Summary (Continued)

Offset Address Register[1]	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
0x07C RAW_DATA REGH1 Page 336	RAW_DATA1(63–48)															
	RAW_DATA1(47–32)															

1 The physical address (base + offset) of these registers is device specific. See the device-specific data sheet to verify the register base address.

9.4.1 CRC Global Control Register 0 (CRC_CTRL0)

Figure 9-3 and Table 9-1 describe this register.

Figure 9-3. CRC Global Control Register 0 (CRC_CTRL0) [offset = 0x00]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															CH1_PSA_SWREST
R-0															R/W-0

R = Read; W = Write; *-n* = Value after reset

Table 9-1. CRC Global Control Register 0 (CRC_CTRL0) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved		These bits are read as zero, and writes have no effect.
0	CH1_PSA_SWREST		Channel 1 PSA software reset: When set, the PSA signature register is reset to all zeros. Software reset does not reset software reset bit itself. Therefore, the CPU is required to clear this bit by writing a 0.
		0	The PSA signature register is not reset.
		1	The PSA signature register is reset.

9.4.2 CRC Global Control Register (CRC_CTRL1)

Figure 9-4 and Table 9-2 describe this register.

Figure 9-4. CRC Global Control (CRC_CTRL1) [offset = 0x08]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															PWDN
R-0															R/W-0

R = Read; W = Write; -*n* = Value after reset

Table 9-2. CRC Global Control (CRC_CTRL1) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved		These bits are read as zero, and writes have no effect.
0	PWDN		Power-down.
		0	CRC is not in power-down mode.
		1	CRC is in power-down mode.

9.4.3 CRC Global Control Register 2 (CRC_CTRL2)

The CRC_CTRL2 register is used to configure the CRC channel in data capture mode. Data capture mode is especially useful when it is used in conjunction with data trace (CH1_TRACEEN) for channel 1. The seed value can be planted in the PSA signature register during data capture mode by writing a seed value into the PSA signature register. The data trace enable bit is then set to snoop and compress any read transactions on the CPU bus. When the trace enable bit is set, CH1_MODE is automatically reset to data capture mode.

To change from one mode to another, the user should perform the following steps:

1. Assert software reset for the respective channel.
2. Change from the current active mode to data capture mode.
3. Change from data capture mode to the new mode.
4. Release the software reset.

Figure 9-5 and Table 9-3 describe this register.

Figure 9-5. CRC Global Control Register 2 (CRC_CTRL2) [offset = 0x10]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											CH1_TRACEEN	Reserved		CH1_MODE	
R-0											R/W-0	R-0		R/W-0	

R = Read; W = Write; -*n* = Value after reset

.

Table 9-3. CRC Global Control Register 2 (CRC_CTRL2) Field Descriptions

Bit	Name	Value	Description
31–5	Reserved		These bits are read as zero, and writes have no effect
4	CH1_TRACEEN		Channel 1 trace enable. When set, the channel is put into data trace mode. The channel snoops on the CPU data bus for any read transactions. Any read data on the bus is compressed by the PSA signature register. When the CPU is in debug state, the PSA signature register does not compress any read data on the bus.
		0	Data trace is disabled.
		1	Data trace is enabled.
3–2	Reserved		These bits are read as zero, and writes have no effect
1–0	CH1_MODE		Channel 1 mode
		00	The CRC is in data capture mode. In this mode, the PSA signature register does not compress data when it is written. Any data written to the PSA signature register is simply captured by the PSA signature register without any compression. This mode can be used to plant a seed value into the PSA register.
		Other	Reserved.

9.4.4 PSA Signature Low Register 1 (PSA_SIGREGL1)

Figure 9-6 and Table 9-4 describe this register.

Figure 9-6. PSA Signature Low Register 1 (PSA_SIGREGL1) [offset = 0x60]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PSASIG1(31–16)															
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSASIG1(15–0)															
R/W-0															

R = Read; W = Write; *-n* = Value after reset

Table 9-4. PSA Signature Low Register 1 (PSA_SIGREGL1) Field Descriptions

Bit	Name	Value	Description
31–0	PSASIG1(31–0)	0–FFFF FFFFh	Channel 1 PSA signature low. These bits contain the PSASIG1(31–0) of the Channel 1 PSA signature.

9.4.5 PSA Signature High Register 1 (PSA_SIGREGH1)

Figure 9-7 and Table 9-5 describe this register.

Figure 9-7. PSA Signature High Register 1 (PSA_SIGREGH1) [offset = 0x64]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PSASIG1(63–48)															
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSASIG1(47–32)															
R/W-0															

R = Read; *-n* = Value after reset

Table 9-5. PSA Signature High Register 1 (PSA_SIGREGH1) Field Descriptions

Bit	Name	Value	Description
31–0	PSASIG1(63–32)	0–FFFF FFFFh	Channel 1 PSA signature high. These bits contain the PSASIG1(63–32) of the Channel 1 PSA signature.

9.4.6 PSA Sector Signature Low Register 1 (PSA_SECSIGREGL1)

Figure 9-8 and Table 9-6 describe this register.

Figure 9-8. PSA Sector Signature Low Register 1 (PSA_SECSIGREGL1) [offset = 0x70]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PSASECSIG1(31–16)															
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSASECSIG1(15–0)															
R-0															

R = Read; W = Write; *-n* = Value after reset

Table 9-6. PSA Sector Signature Low Register 1 (PSA_SECSIGREGL1) Field Descriptions

Bit	Name	Value	Description
31–0	PSASECSIG1(31–0)	0–FFFF FFFFh	Channel 1 PSA sector signature low. These bits contain the value stored in the PSASECSIG1(31–0) register.

9.4.7 PSA Sector Signature High Register 1 (PSA_SECSIGREGH1)

Figure 9-9 and Table 9-7 describe this register.

Figure 9-9. PSA Sector Signature High Register 1 (PSA_SECSIGREGH1) [offset = 0x74]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PSASECSIG1(63–48)															
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSASECSIG1(47–32)															
R-0															

R = Read; W = Write; *-n* = Value after reset

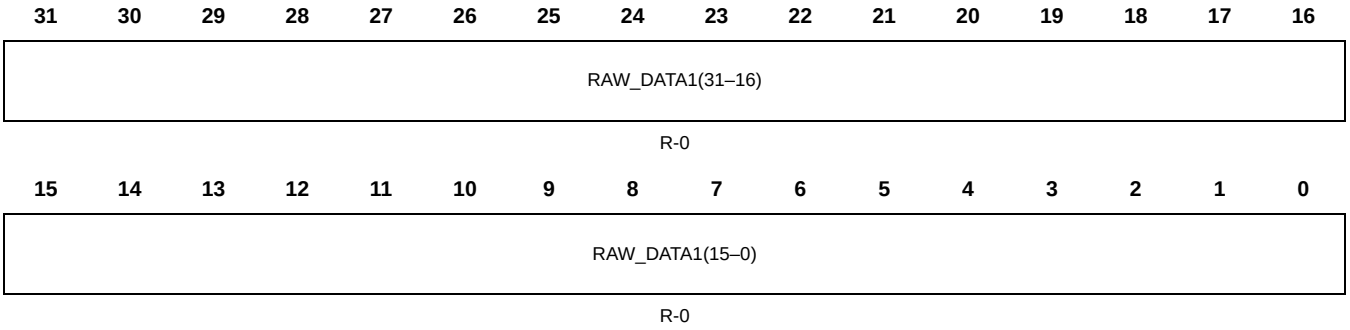
Table 9-7. PSA Sector Signature High Register 1 (PSA_SECSIGREGH1) Field Descriptions

Bit	Name	Value	Description
31–0	PSASECSIG1(63–32)	0–FFFF FFFFh	Channel 1 PSA sector signature high. These bits contain the value stored in the PSASECSIG1(63–32) register.

9.4.8 Raw Data Low Register 1 (RAW_DATAREGL1)

Figure 9-10 and Table 9-8 describe this register.

Figure 9-10. Raw Data Low Register 1 (RAW_DATAREGL1) [offset = 0x78]



R = Read; W = Write; *-n* = Value after reset

Table 9-8. Raw Data Low Register 1 (RAW_DATAREGL1) Field Descriptions

Bit	Name	Value	Description
31–0	RAW_DATA1(31–0)	0–FFFF FFFFh	Channel 1 raw data low. These bits contain bits 31–0 of the uncompressed raw data.

9.4.9 Raw Data High Register 1 (RAW_DATAREGH1)

Figure 9-11 and Table 9-9 describe this register.

Figure 9-11. Raw Data High Register 1 (RAW_DATAREGH1) [offset = 7Ch)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RAW_DATA1(63–48)															
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RAW_DATA1(47–32)															
R-0															

R = Read; W = Write; *-n* = Value after reset

Table 9-9. Raw Data High Register 1 (RAW_DATAREGH1) Field Descriptions

Bit	Name	Value	Description
31–0	RAW_DATA1(63–32)		Channel 1 raw data high. These bits contain bits 63–32 of the uncompressed raw data.

Controller Area Network (DCAN)

Topic		Page
10.1	Overview	338
10.2	Operating Modes	341
10.3	Dual Clock Source	347
10.4	GIO support	348
10.5	RAM Initialization	349
10.6	Interrupt functionality	350
10.7	Global power down mode	351
10.8	Local power down mode	352
10.9	Parity Check Mechanism	354
10.10	Debug/Suspend Mode	355
10.11	Module Initialization	356
10.12	Configuration of Message Objects	357
10.13	Message Handling	360
10.14	CAN Bit Timing	365
10.15	Message Interface Register Sets	374
10.16	Message RAM	377
10.17	DCAN Control Registers	385

10.1 Overview

This reference guide contains a general description of the DCAN Controller Area Network module. The Controller Area Network is a serial communications protocol which efficiently supports distributed real-time control with a high level of security.

The DCAN module supports bit-rates up to 1 Mbit/s and is compliant to the CAN 2.0B protocol specification.

10.1.1 Features

The DCAN implements the following features:

- Supports CAN protocol version 2.0 part A, B
- Bit rates up to 1 MBit/s
- Dual clock source
- 16, 32, 64 or 128 message objects (device dependent)
- Individual identifier mask for each message object
- Programmable FIFO mode for message objects
- Programmable loop-back modes for self-test operation
- Suspend mode for debug support
- Software module reset
- Automatic bus on after Bus-Off state by a programmable 32-bit timer
- Message RAM parity check mechanism
- Direct access to Message RAM during test mode
- CAN Rx / Tx pins configurable as general purpose IO pins
- Two interrupt lines
- Global power down and wakeup support
- Local power down and wakeup support
- RAM initialization

10.1.2 Functional Description

The DCAN performs CAN protocol communication according to ISO 11898-1 (identical to Bosch CAN protocol specification 2.0 A, B). The bit rate can be programmed to values up to 1 MBit/s. Additional transceiver hardware is required for the connection to the physical layer (CAN bus).

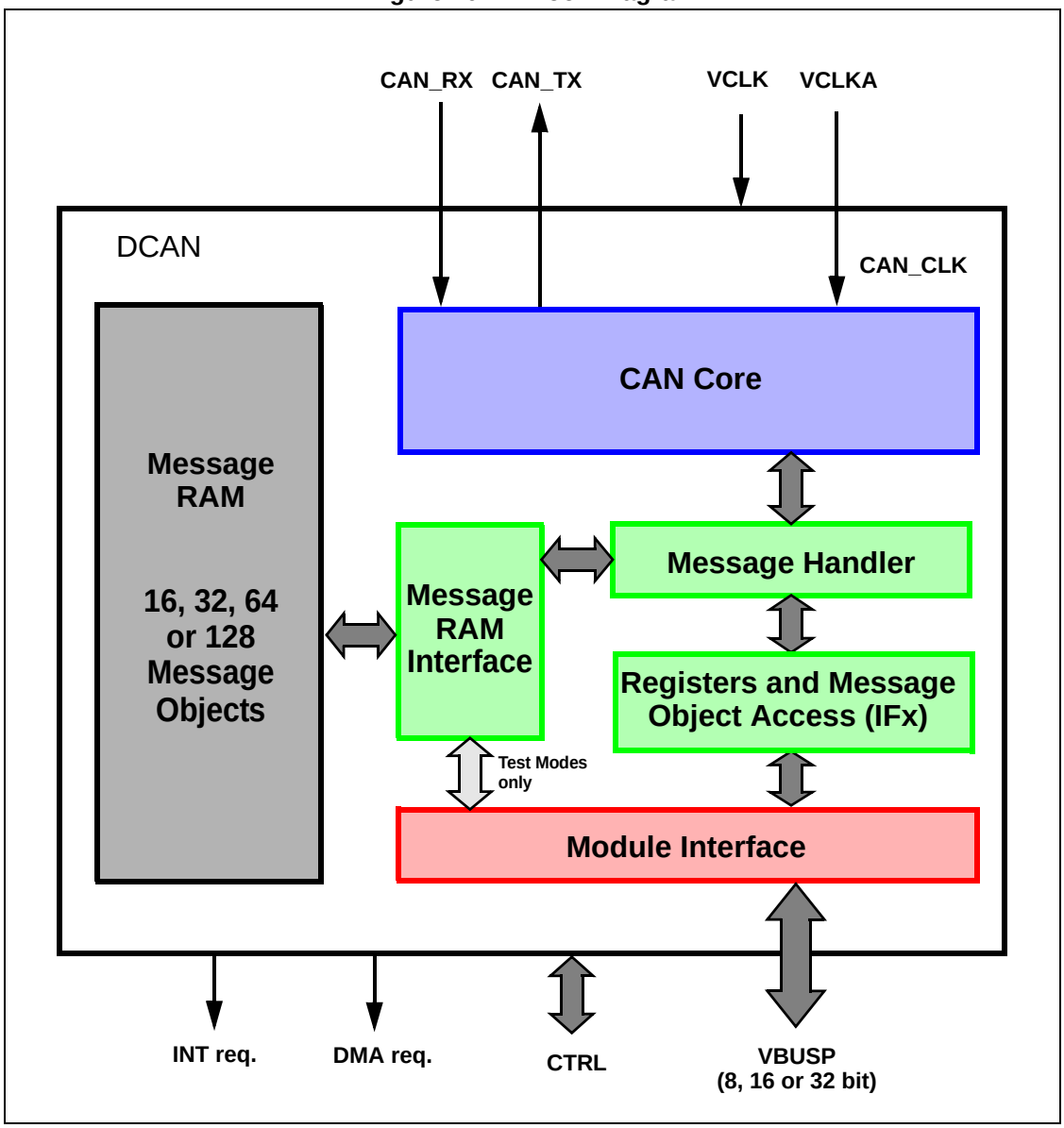
For communication on a CAN network, individual message objects can be configured. The message objects and identifier masks are stored in the Message RAM.

All functions concerning the handling of messages are implemented in the Message Handler. Those functions are acceptance filtering, the transfer of messages between the CAN Core and the Message RAM, the handling of transmission requests as well as the generation of interrupts or DMA requests.

The register set of the DCAN can be accessed directly by the CPU via the module interface. These registers are used to control/configure the CAN Core and the Message Handler, and to access the Message RAM.

10.1.3 Block Diagram

Figure 10-1. Block Diagram



10.1.3.1 CAN Core

The CAN Core consists of the CAN Protocol Controller and the Rx/Tx Shift Register. It handles all ISO 11898-1 protocol functions.

10.1.3.2 Message Handler

The Message Handler is a state machine which controls the data transfer between the single ported Message RAM and the CAN Core's Rx/Tx Shift Register. It also handles acceptance filtering and the interrupt/DMA request generation as programmed in the control registers.

10.1.3.3 Message RAM

The DCAN Message RAM enables the storage of 16, 32, 64 or 128 CAN messages (device dependent).

10.1.3.4 Registers and Message Object Access (IFx)

Data consistency is ensured by indirect accesses to the message objects. During normal operation, all CPU and DMA accesses to the Message RAM are done through Interface Registers.

Three Interface Register sets control the CPU read and write accesses to the Message RAM. There are two Interface Registers sets for read / write access (IF1 and IF2) and one Interface Register set for read access only (IF3). See also section 10.15. The Interface Registers have the same word-length as the Message RAM.

In a dedicated test mode, the Message RAM is memory mapped and can be directly accessed.

10.1.3.5 Module Interface

The DCAN module is equipped with a 32-bit interface (VBUSP).

10.1.3.6 Dual Clock Source

Two clock domains are provided to the DCAN module: the peripheral synchronous clock domain (VCLK) and the peripheral asynchronous clock source domain (VCLKA) for CAN_CLK.

If a frequency modulated clock output from FMPLL is used as the VCLK source, then VCLKA should be derived from an unmodulated clock source (e.g. OSCIN source). See also section 10.3.

The clock source for VCLKA is selected by the Peripheral Asynchronous Clock Source Register in the system module. For more information see the TMS470M Platform Architecture Specification chapter.

10.2 Operating Modes

10.2.1 Software Initialization

The software initialization mode is entered by setting the **Init** bit in the CAN Control Register, either by software, by hardware reset or by going Bus-Off.

While the Init bit is set, the message transfer from and to the CAN bus is stopped, and the status of the CAN_TX output is recessive (high). The CAN error counters are not updated. Setting the Init bit does not change any other configuration register.

To initialize the CAN Controller, the CPU has to set up the CAN Bit timing and those message objects which have to be used for CAN communication. Message objects which are not needed, can be deactivated by with their **MsgVal** bits cleared (default after RAM initialization).

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.

Resetting the Init bit finishes the software initialization. Afterwards the Bit Stream Processor (BSP, for more details see section 10.14) synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (= Bus Idle) before it can take part in bus activities and start the message transfer.

The initialization of the message objects is independent of the Init bit, however all message objects should be configured to particular identifiers or set to not valid before the message transfer is started.

It is possible to change the configuration of message objects during normal operation by the CPU. After setup and subsequent transfer of message object from Interface Registers to Message RAM, the acceptance filtering will be applied to it, when the modified message object number is same or smaller than the previously found message object. This assures data consistency, even when changing message objects e.g. while a pending CAN frame reception.

10.2.2 CAN Message Transfer (Normal Operation)

Once the DCAN is initialized and **Init** bit is reset to zero, the CAN Core synchronizes itself to the CAN bus and starts the message transfer.

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 e.g. the identifier mask is used, the arbitration bits which are masked to “don’t care” may change in the message object when a received message is stored.

The CPU may read or write each message at any time via the Interface Registers, as the Message Handler guarantees data consistency in case of concurrent accesses.

Messages to be transmitted can be updated by the CPU. If a permanent message object (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. 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.

10.2.2.1 Disabled Automatic Retransmission

According to the CAN Specification (see ISO11898, 6.3.3 Recovery Management), the DCAN provides a mechanism to automatically retransmit frames which have lost arbitration or have been disturbed by errors during transmission. The frame transmission service will not be confirmed to the user before the transmission is successfully completed.

By default, this automatic retransmission is enabled. It can be disabled by setting bit **DAR** in CAN Control Register. Further details to this mode are provided in section 10.13.3.

10.2.2.2 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 which is enabled by bit **ABO** in CAN Control Register (see section 10.17.1). 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 which can be defined in Auto-Bus-On Time Register (see section 10.17.9).

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.

10.2.3 Test Modes

The DCAN provides several test modes which are mainly intended for production tests or self test.

For all test modes, bit **Test** in the CAN Control Register needs to be set to one. This enables write access to the Test Register (see section 10.17.6).

10.2.3.1 Silent Mode

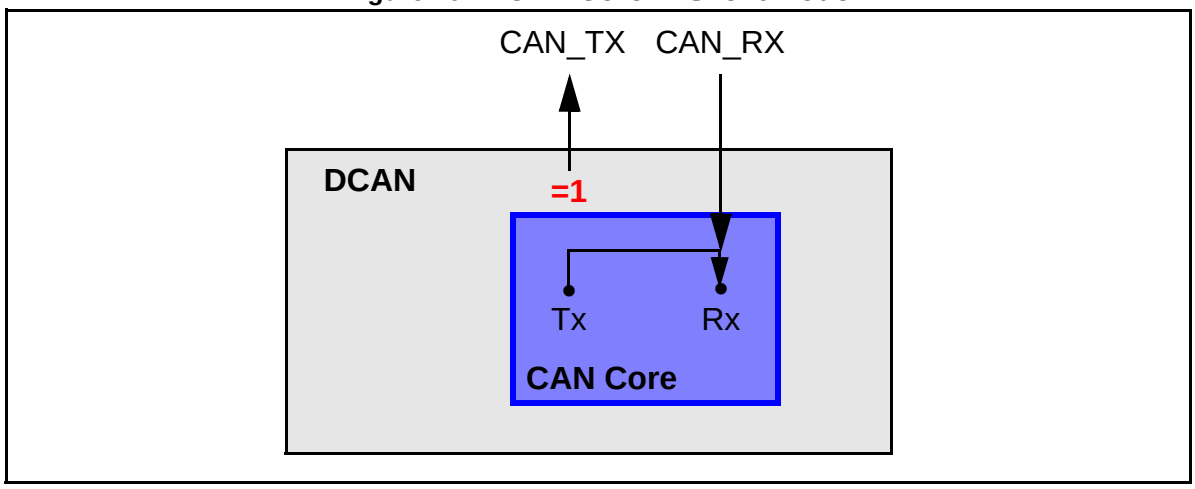
The Silent Mode may be used to analyze the traffic on the CAN bus without affecting it by sending dominant bits (e.g. acknowledge bit, overload flag, active error flag). The DCAN is still able to receive valid data frames and valid remote frames, but it will not send any dominant bits. However, these are internally routed to the CAN Core.

Figure 10-2 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 one.

In ISO 11898-1, the Silent Mode is called the Bus Monitoring Mode.

Figure 10-2. *CAN Core in Silent Mode*



10.2.3.2 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.

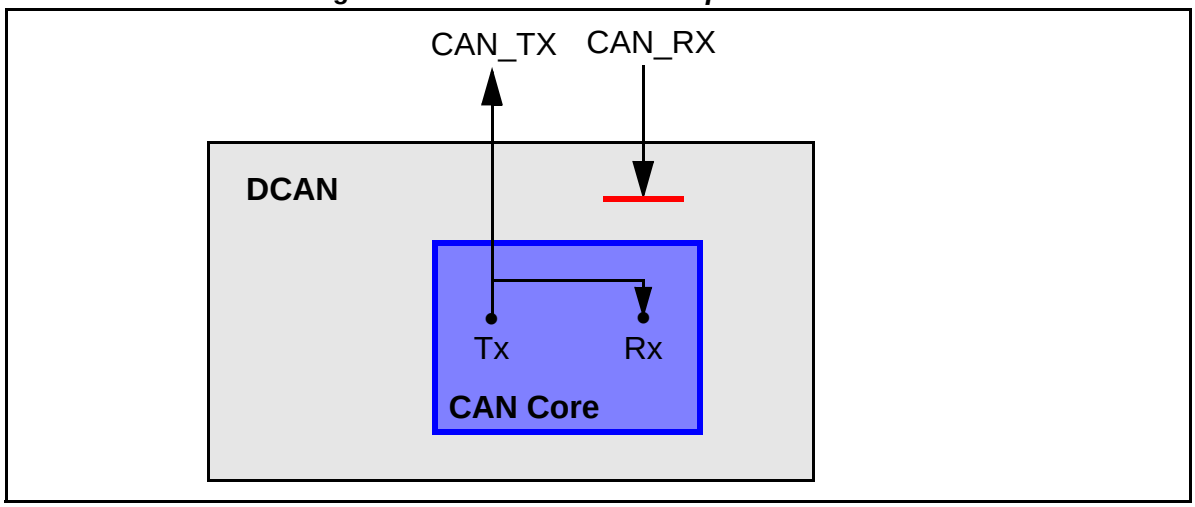
In order to be independent from external stimulation, the CAN Core ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in Loop Back Mode.

Figure 10-3 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 bit **LBack** in Test Register to one.

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 10.2.3.3).

Figure 10-3. *CAN Core in Loop Back Mode*



10.2.3.3 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.

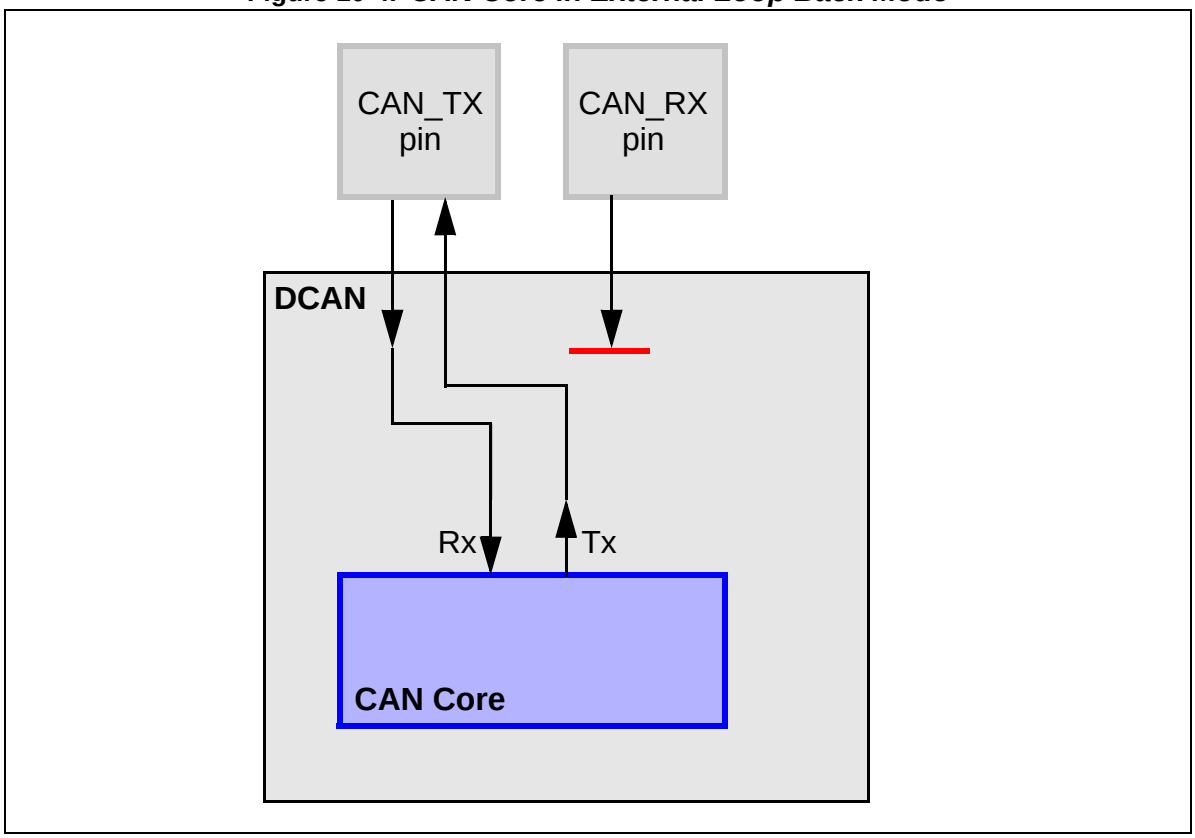
With this configuration, the Tx pin IO circuit can be tested.

External Loop Back Mode can be activated by setting bit **ExL** in Test Register to one.

Figure 10-4 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 set), the ExL bit will be ignored.

Figure 10-4. *CAN Core in External Loop Back Mode*

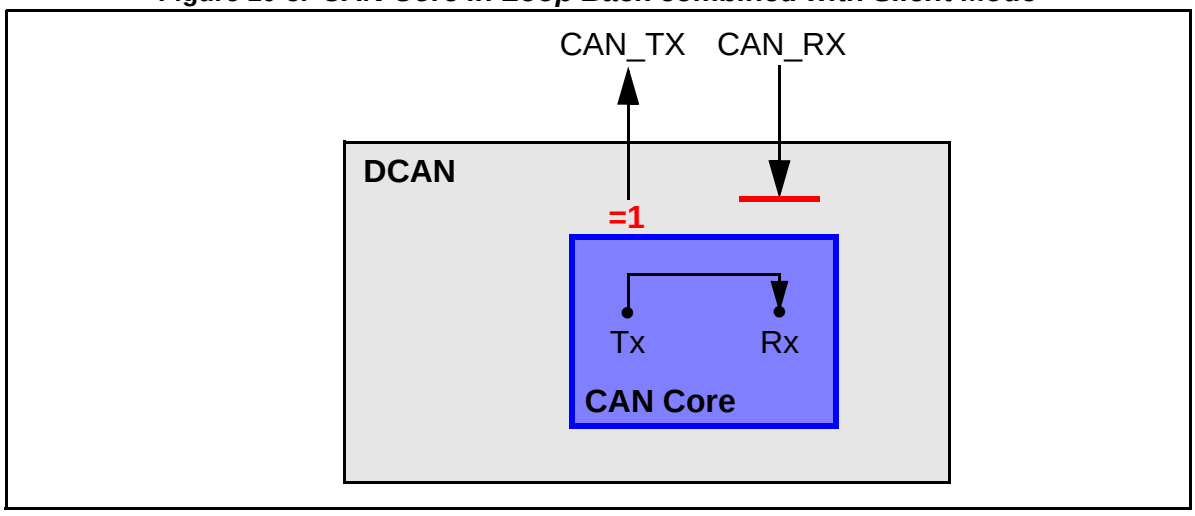


10.2.3.4 Loop Back combined with Silent Mode

It is also possible to combine Loop Back Mode and Silent Mode by setting bits **LBack** and **Silent** at the same time. This mode can be used for a “Hot Selftest”, i.e. 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 10-5 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 10-5. *CAN Core in Loop Back combined with Silent Mode*



10.2.3.5 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 bits Tx[1:0] as described in section 10.17.6.

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.

10.3 Dual Clock Source

Two clock domains are provided to the DCAN module: the peripheral synchronous clock domain (VCLK) as the general module clock source, and the peripheral asynchronous clock source domain (VCLKA) provided to the CAN core (as clock source CAN_CLK) for generating the CAN Bit Timing.

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 (e.g. derived directly from the OSCIN clock) should be used for VCLKA.

Please see 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:

$$f_{0,\,VCLK} \pm \Delta f_{FM,\,VCLK} \geq f_{CANCLK}$$

Note:
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.

10.4 GIO support

The CAN_RX and CAN_TX pins of the 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 10.17.30) and the CAN RX IO Control register (see section 10.17.31).

10.5 RAM Initialization

If the memory hardware initialization for the DCAN module is enabled in the device system module control registers, with a dedicated action the complete RAM can be initialized with zeros and the parity bits are set accordingly. For more details, please refer to the system module reference guide.

Initialization speed of DCAN RAM:

(# of message objects + 5) * VBUS cycle time

On power up, the RAM should be initialized by activating global SoC signal **CAN_MMISTART**. This initializes the RAM with zeros and sets the parity/ECC bits accordingly. The first access over VBUSP should not be started before RAM initialization has finished, indicated by the DCAN acknowledge signal **CAN_MMIDONE**. Accesses while or before RAM initialization are not allowed and could result in parity errors or other side effects.

The initialization speed is one Message Object per **VBUSP_CLK** cycle plus approximate 5 **VBUSP_CLK** cycles internal runtime till asserting **CAN_MMIDONE**.

10.6 Interrupt functionality

Interrupts can be generated on two interrupt lines in the: DCAN0INT line and DCAN1INT line. These lines can be enabled by setting IE0 resp. IE1 bits in CAN Control Register.

The DCAN provides three groups of interrupt sources: Message Object Interrupts, Status Change Interrupts and Error Interrupts.

The source of an interrupt can be determined by the interrupt identifiers Int0ID / Int1ID in the Interrupt Register (see section 10.17.5). 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 (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, 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 which reads the message that is the source of the interrupt, may read the message and reset the message object's IntPnd at the same time (ClrIntPnd bit in the IF1/IF2 Command Register). When IntPnd is cleared, the Interrupt Register will point to the next message object with a pending interrupt.

10.6.1 Message Object interrupts

Message Object interrupts are generated by events from the message objects. They are controlled by the flags IntPND, TxIE and RxIE which are described in section 10.16.1.

Message Object interrupts can be routed to either DCAN0INT or DCAN1INT line, controlled by the Interrupt Multiplexer Register, see section 10.17.18.

10.6.2 Status Change Interrupts

The events WakeUpPnd, RxOk, TxOk and LEC in Error and Status Register (see section 10.17.2) belong to the Status Change Interrupts. The Status Change Interrupt group can be enabled by bit SIE in CAN Control Register (see section 10.17.1).

If SIE is set, a Status Change Interrupt will be generated at each CAN frame, independent of bus errors or valid CAN communication, and also independent of the Message RAM configuration.

Status Change interrupts can only be routed to interrupt line DCAN0INT which has to be enabled by setting IE0 in the CAN Control Register.

Note:
Reading the Error and Status Register will clear the WakeUpPnd flag. If in global power down mode, the WakeUpPnd flag is cleared by such a read access before the DCAN module has been waken up by the system, the DCAN may re-assert the WakeUpPnd flag, and a second interrupt may occur. (see also section 10.7.2).

10.6.3 Error Interrupts

The events PER, BOff and EWarn (monitored in Error and Status Register, see section 10.17.2) belong to the Error Interrupts. The Error Interrupt group can be enabled by setting bit EIE in CAN Control Register (see section 10.17.1).

Error interrupts can only be routed to interrupt line DCAN0INT which has to be enabled by setting IE0 in the CAN Control Register.

10.7 Global power down mode

The TMSx70 architecture supports a centralized global power down control over the peripheral modules through the Peripheral Central Resource (PCR) module (see also TMS470 Platform Architecture Specification).

10.7.1 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.

During local power down mode, the internal clocks of the DCAN module are turned off, but there is a wake up logic that is still active. Also the actual contents of the DCAN control registers can be read back.

10.7.2 Wakeup from global power down mode

If the DCAN module is in global power down mode, a CAN bus activity detection circuit is active. On occurrence of a dominant CAN bus level, the DCAN will set the **WakeUpPnd** bit in Error and Status Register (see section 10.17.2).

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.

10.8 Local power down mode

Besides from the centralized power down mechanism controlled by the PCR module (global power down, see section 10.7), the DCAN supports a local power down mode which can be controlled within the DCAN control registers.

10.8.1 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 10.8.2) which 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.

10.8.2 Wakeup from local power down

There are two ways to wake up the DCAN from local power down mode:

The application could wake up the DCAN module manually by clearing the PDR bit and then clearing the Init bit in CAN Control Register.

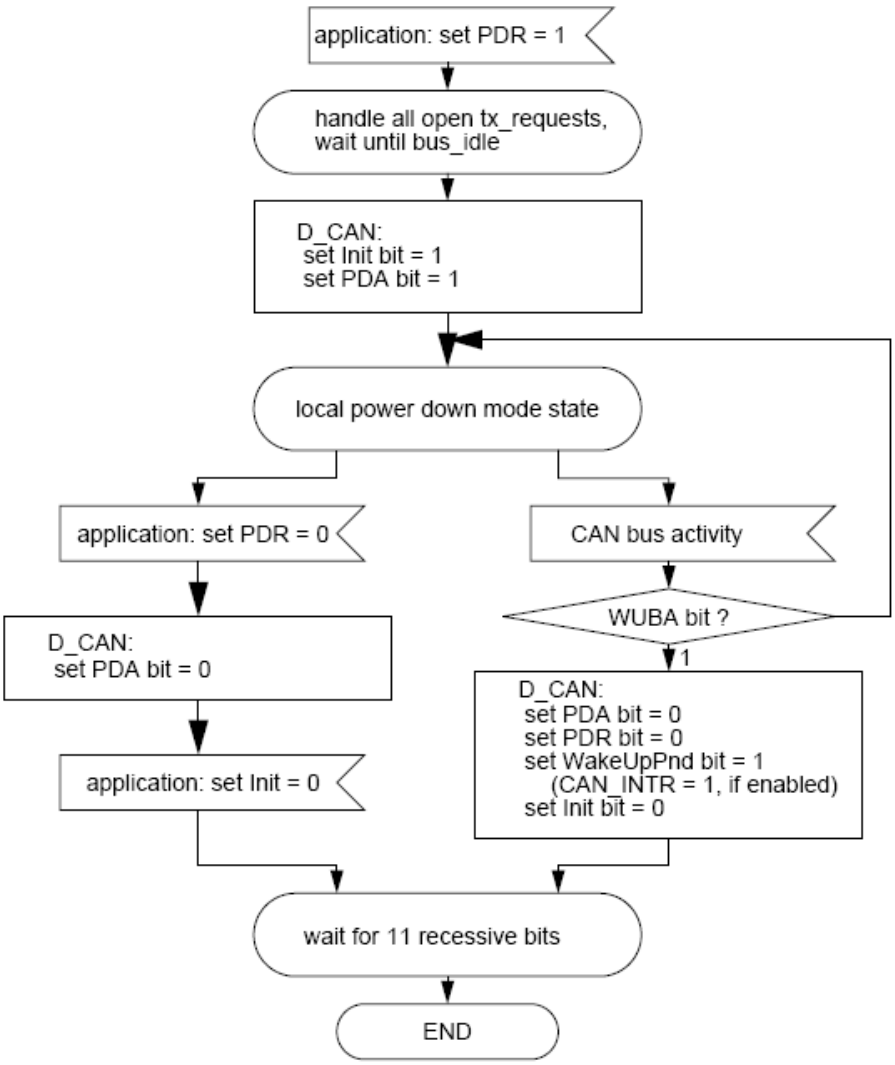
Alternatively, a CAN bus activity detection circuit can be activated by setting the wake up on bus activity bit (**WUBA**) in CAN Control Register. If this circuit is active, on occurrence of a dominant CAN bus level, the DCAN will automatically start the wake up sequence. It will clear the PDR bit in 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 10-6 shows a flow diagram about entering and leaving local power down mode.

Figure 10-6. Local power down mode flow diagram



10.9 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 10.16.4.

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, see section 10.17.1.

In case of disabled parity check, the parity bits in message RAM will be left unchanged on write access to data area and no check will be done on read access.

If parity checking is enabled, parity bits will be automatically generated and checked by the DCAN. The parity bits could be read in Debug/Suspend mode (see section 10.16.3) or in RDA mode (see section 10.16.4). 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.

10.9.1 Behavior on parity error

On any read access to Message RAM, e.g. 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 **MsgVal** 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, e.g. by immediately checking the Parity Error Code register on parity error interrupt.

Note:
During RAM initialization, no parity check will be done.

10.9.2 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.

10.10 Debug/Suspend Mode

The module supports the usage of an external debug unit by providing functions like pausing DCAN activities and making Message RAM content accessible via VBUSP interface.

Before entering debug/suspend mode, the circuit will either wait until a started transmission or reception will be finished and Bus idle state is recognized, or immediately interrupt a current transmission or reception. This is depending on bit **IDS** in CAN Control Register (see section 10.17.1).

Afterwards, the DCAN enters debug/suspend mode, indicated by **InitDbg** flag in CAN Control Register.

During Debug/Suspend mode, all DCAN registers can be accessed. Reading reserved bits will return '0'. Writing to reserved bits will have no effect.

Also, the Message RAM will be memory mapped. This allows the external debug unit to read the Message RAM. For the memory organization, see section 10.16.3).

Note:
During Debug/Suspend Mode, the Message RAM cannot be accessed via the IFx register sets.

Note:
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 DMAActive flag by r/w)

10.11 Module Initialization

After hardware reset, the Init bit in the CAN Control Register is set and all CAN protocol functions are disabled. The configuration of the Bit timing and of the message objects should be completed before the CAN protocol functions are enabled.

For the configuration of the message objects, see section 10.12.

For the configuration of the Bit Timing, see section 10.14.2.

The bits MsgVal, NewDat, IntPnd, and TxRqst of the message objects are reset to ‘0’ by a hardware reset. The configuration of a message object is done by programming Mask, Arbitration, Control and Data bits of one of the IF1/IF2 Interface Register sets to the desired values. By writing the message object number to bits [7:0] of the corresponding IF1/IF2 Command Register, the IF1/IF2 Interface Register content is loaded into the addressed message object in the Message RAM.

It is possible to initialize all message objects in the Message RAM by memory hardware initialization (triggered by the device’s system module, see system module reference guide for details).

The configuration of the bit timing requires that the CCE bit in the CAN Control Register is set additionally to Init. This is not required for the configuration of the message objects.

When the Init bit in the CAN Control Register is cleared, the CAN Protocol Controller state machine of the CAN Core and the Message Handler State Machine start to control the DCAN’s internal data flow. Received messages which pass the acceptance filtering are stored into the Message RAM; messages with pending transmission request are loaded into the CAN Core’s Shift Register and are transmitted via the CAN bus.

The CPU may enable the interrupt lines (setting IE0 and IE1 to ‘1’) at the same time when it clears Init and CCE. The status interrupts EIE and SIE may be enabled simultaneously.

The CAN communication may be controlled interrupt-driven or in polling mode. The Interrupt Register points to those message objects with IntPnd = ‘1’. It is updated even if the interrupt lines to the CPU are disabled (IE0 / IE1 are zero).

The CPU may poll all MessageObject’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.

10.12 Configuration of Message Objects

The whole Message RAM should to be configured before the end of the initialization, however it is also possible to change the configuration of message objects during CAN communication.

The CAN software driver library should offer subroutines that:

- Transfer a complete message structure into a message object. (Configuration)
- Transfer the data bytes of a message into a message object and set TxRqst and NewDat. (Start a new transmission)
- Get the data bytes of a message from a message object and clear NewDat (and IntPnd). (Read received data)
- Get the complete message from a message object and clear NewDat (and IntPnd). (Read a received message, including identifier, from a message object with UMask = '1')

Parameters of the subroutines are the Message Number and a pointer to a complete message structure or to the data bytes of a message structure.

Two examples of assigning the IFx Interface Register sets to these subroutines are shown here:

In the first method, the tasks of the application program that may access the module are divided in two groups. Each group is restricted to the use of one of the Interface Register sets. The tasks of one group may interrupt tasks of the other group, but not of the same group.

In the second method, which may be a special case of the first method, there are only two tasks in the application program that access the module. A Read_Message task that uses IF2 or IF3 to get received messages from the Message RAM and a Write_Message task that uses IF1 to write messages to be transmitted (or to be configured) into the Message RAM. Both tasks may interrupt each other.

10.12.1 Configuration of a Transmit Object for Data Frames

Figure 10-7 shows how a Transmit Object can be initialized.

Figure 10-7. Initialization of a Transmit Object

MsgVal	Arb	Data	Mask	EoB	Dir	NewDat	MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
1	appl.	appl.	appl.	1	1	0	0	0	appl.	0	appl.	0

The Arbitration bits (ID[28:0] and Xtd bit) are given by the application. They define the identifier and type of the outgoing message. If an 11-bit Identifier (Standard Frame) is used (Xtd = '0'), it is programmed to ID[28:18]. In this case, ID[17:0] can be ignored.

The Data Registers (DLC[3:0] and Data0-7) are given by the application, TxRqst and RmtEn should not be set before the data is valid.

If the TxIE bit is set, the IntPnd bit will be set after a successful transmission of the message object.

If the RmtEn bit is set, a matching received Remote Frame will cause the TxRqst bit to be set; the 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 10.13.8. Identifier masking must be disabled (UMask = '0') if no Remote Frames are allowed to set the TxRqst bit (RmtEn = '0').

10.12.2 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.

10.12.3 Configuration of a Single Receive Object for Data Frames

Figure 10-8 shows how a Receive Object for Data Frames can be initialized.

Figure 10-8. Initialization of a single Receive Object for Data Frames

MsgVal	Arb	Data	Mask	EoB	Dir	NewDat	MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
1	appl.	appl.	appl.	1	0	0	0	appl.	0	0	0	0

The Arbitration bits (ID[28:0] and Xtd bit) are given by the application. They define the identifier and type of accepted received messages. If an 11-bit Identifier (Standard Frame) is used (Xtd = '0'), it is programmed to ID[28:18]. In this case, ID[17:0] can be ignored. When a Data Frame with an 11-bit Identifier is received, ID[17:0] will be set to '0'.

The Data Length Code (DLC[3:0]) is given by the application. When the Message Handler stores a Data Frame in the message object, it will store the received Data Length Code and eight data bytes. If the Data Length Code is less than 8, the remaining bytes of the message object may be overwritten by non specified values.

The Mask bits (Msk[28:0], UMask, MXtd, and MDir bits) may be used (UMask = '1') to allow groups of Data Frames with similar identifiers to be accepted. The Dir bit should not be masked in typical applications. If some bits of the Mask bits are set to "don't care", the corresponding bits of the Arbitration Register will be overwritten by the bits of the stored Data Frame.

If the RxIE bit is set, the IntPnd bit will be set when a received Data Frame is accepted and stored in the message object.

If the TxRqst bit is set, the transmission of a Remote Frame with the same identifier as actually stored in the Arbitration bits will be triggered. The content of the Arbitration bits may change if the Mask bits are used (UMask = '1') for acceptance filtering.

10.12.4 Configuration of a Single Receive Object for Remote Frames

Figure 10-9 shows how a Receive Object for Remote Frames can be initialized.

Figure 10-9. Initialization of a single Receive Object for Remote Frames

MsgVal	Arb	Data	Mask	EoB	Dir	NewDat	MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
1	appl.	appl.	appl.	1	1	0	0	appl.	0	0	0	0

Receive Objects for Remote Frames may be used to monitor Remote Frames on the CAN bus. The Remote Frame stored in the Receive Object will not trigger the transmission of a Data Frame. Receive Objects for Remote Frames may be expanded to a FIFO buffer, see section 10.12.5.

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 10.13.8.

The Arbitration bits (ID[28:0] and Xtd bit) may be given by the application. They define the identifier and type of accepted received Remote Frames. If some bits of the Mask bits are set to "don't care", the corresponding bits of the Arbitration bits will be overwritten by the bits of the stored Remote Frame. If an 11-bit Identifier (Standard Frame) is used (Xtd = '0'), it is programmed to ID[28:18]. In this case, ID[17:0] can be ignored. When a Remote Frame with an 11-bit Identifier is received, ID[17:0] will be set to '0'.

The Data Length Code (DLC[3:0]) may be given by the application. When the Message Handler stores a 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.

10.12.5 Configuration of a FIFO Buffer

With the exception of the EoB bit, the configuration of Receive Objects belonging to a FIFO Buffer is the same as the configuration of a single Receive Object.

To concatenate multiple message objects to a FIFO Buffer, the identifiers and masks (if used) of these message objects have to be programmed to matching values. Due to the implicit priority of the message objects, the message object with the lowest number will be the first message object of the FIFO Buffer. The EoB bit of all message objects of a FIFO Buffer except the last one have to be programmed to zero. The EoB bits of the last message object of a FIFO Buffer is set to one, configuring it as the end of the block.

10.13 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 stored those frames that are accepted into the designated message objects. The application has to update the data of the messages to be transmitted and to enable and request their transmission. The transmission is requested automatically when a matching Remote Frame is received.

The application may read messages which are received and accepted. Messages that are not read before the next messages is accepted for the same message object will be overwritten.

Messages may be read interrupt-driven or after polling of NewDat.

10.13.1 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 above listed status information of each message object via IFx registers separately, these Message Handler registers provides a fast and easy way to get an overview e.g. about all pending transmission requests.

All Message Handler registers are read-only.

10.13.2 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 e.g. 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.

10.13.3 Transmission of Messages in Event Driven CAN Communication

If the shift register of the CAN Core is ready for loading and if there is no data transfer between the IFx Registers and Message RAM, the MsgVal bits in the Message Valid Register and the TxRqst bits in the Transmission Request Register are evaluated. The valid message object with the highest priority pending transmission request is loaded into the shift register by the Message Handler and the transmission is started. The message object's NewDat bit is reset.

After a successful transmission and if no new data was written to the message object (NewDat = '0') since the start of the transmission, the TxRqst bit will be reset. If TxIE is set, IntPnd will be set after a successful transmission. If the 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.

10.13.4 Updating a Transmit Object

The CPU may update the data bytes of a Transmit Object any time via the IF1/IF2 Interface Registers, neither MsgVal nor TxRqst have to be reset before the update.

Even if only a part of the data bytes are to be updated, all four bytes in the corresponding IF1/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 10.13.3.

When NewDat is set together with TxRqst, NewDat will be reset as soon as the new transmission has started.

10.13.5 Changing a Transmit Object

If the number of implemented message objects is not sufficient to be used as permanent message objects only, the Transmit Objects may be managed dynamically. The CPU can write the whole message (Arbitration,

Control, and Data) into the Interface Register. The bits [23:16] of the Command Register can be set to 0xB7 for the transfer of the whole message object content into the message object. Neither MsgVal nor TxRqst have to be reset before this operation.

If a previously requested transmission of this message object is not completed but already in progress, it will be continued; however it will not be repeated if it is disturbed.

To only update the data bytes of a message to be transmitted, bits [23:16] of the Command Register should be set to 0x87.

Note:
After the update of the Transmit Object, the Interface Register set will contain a copy of the actual contents of the object, including the part that had not been updated.

10.13.6 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.

10.13.7 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.

10.13.8 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.

10.13.9 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.

10.13.10Requesting 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.

10.13.11Storing 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.

10.13.12Reading from a FIFO Buffer

Several messages may be accumulated in a set of message objects which are concatenated to form a FIFO Buffer before the application program is required (in order to avoid the loss of data) to empty the buffer.

A FIFO Buffer of length N will store N-1 plus the last received message since last time it was cleared.

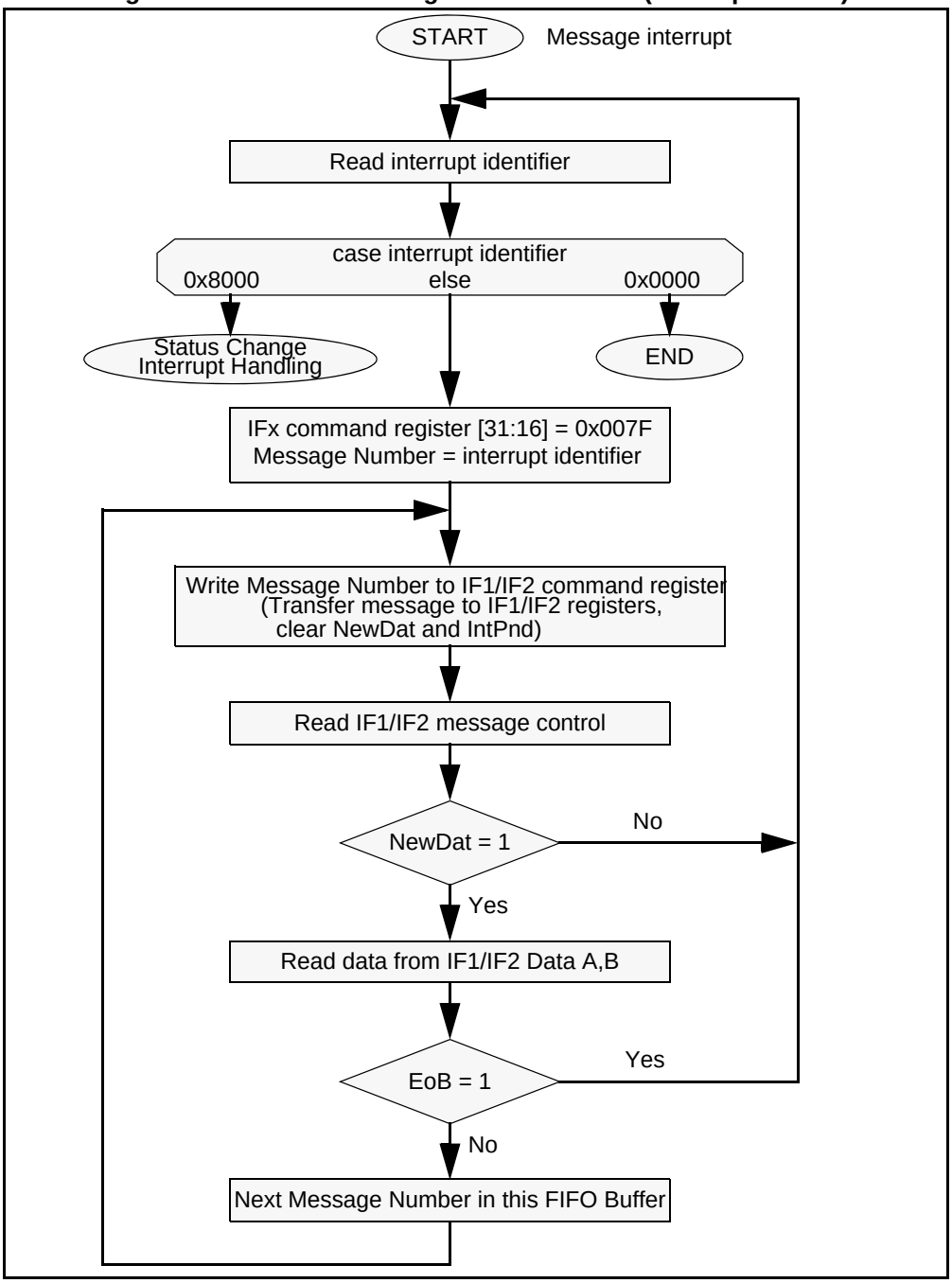
A FIFO Buffer is cleared by reading and resetting the NewDat bits of all its message objects, starting at the FIFO Object with the lowest message number. This should be done in a subroutine following the example shown in Figure 10-10.

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 10-10. CPU Handling of a FIFO Buffer (Interrupt Driven)



10.14 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.

The frequencies of these oscillators are not absolutely stable. Small variations are caused by changes in temperature or voltage and by deteriorating components. As long as the variations remain inside a specific oscillator tolerance range (df), the CAN nodes are able to compensate for the different bit rates by resynchronizing to the bit stream.

In many cases, the CAN bit synchronization will amend a faulty configuration of the CAN bit timing to such a degree that only occasionally an error frame is generated. In the case of arbitration however, when two or more CAN nodes simultaneously try to transmit a frame, a misplaced sample point may cause one of the transmitters to become error passive.

The analysis of such sporadic errors requires a detailed knowledge of the CAN bit synchronization inside a CAN node and of the CAN nodes' interaction on the CAN bus.

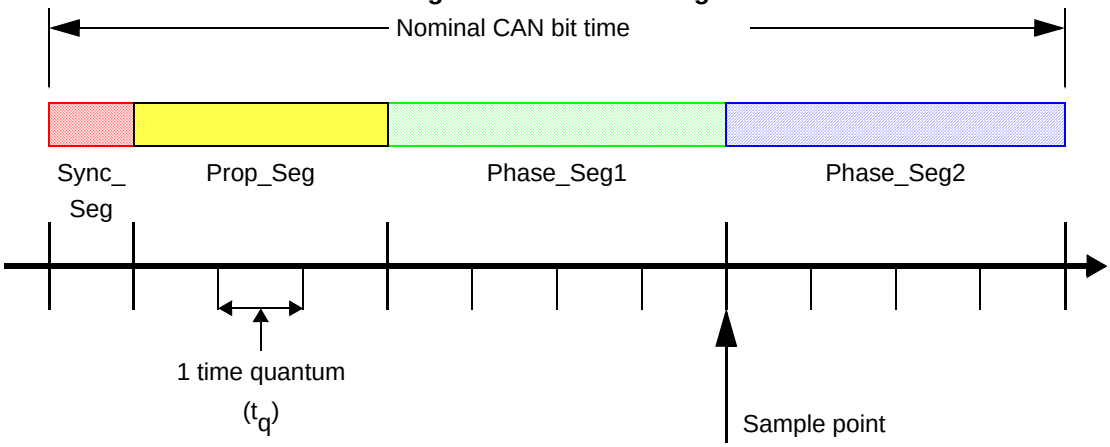
Even if minor errors in the configuration of the CAN bit timing do not result in immediate failure, the performance of a CAN network can be reduced significantly.

10.14.1 Bit Time and Bit Rate

According to the CAN specification, the Bit time is divided into four segments (see Figure 10-11):

- Synchronization Segment (Sync_Seg)
- Propagation Time Segment (Prop_Seg)
- Phase Buffer Segment 1 (Phase_Seg1)
- Phase Buffer Segment 2 (Phase_Seg2)

Figure 10-11. Bit Timing



Each segment consists of a specific number of time quanta. The length of one time quantum (t_q), which is the basic time unit of the bit time, is given by the CAN_CLK and the Baud Rate Prescalers (BRPE and BRP). With these two Baud Rate Prescalers combined, divider values from 1 to 1024 can be programmed:

t_q = Baud Rate Prescaler / CAN_CLK

Apart from the fixed length of the synchronization segment, these numbers are programmable. Table 10-1 describes the minimum programmable ranges required by the CAN protocol.

A given bit rate may be met by different Bit time configurations.

Table 10-1. 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

Note:
For proper functionality of the CAN network, the physical delay times and the oscillator's tolerance range have to be considered.

10.14.1.1 Synchronization Segment

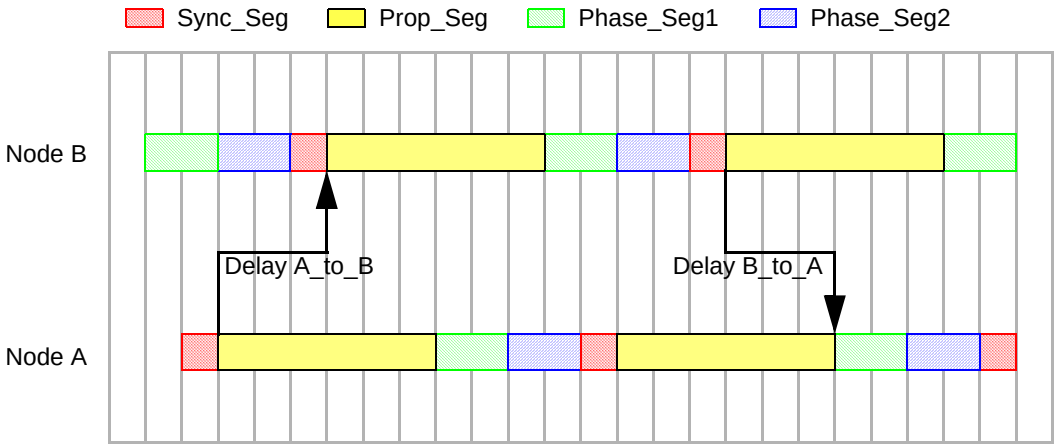
The Synchronization Segment (Sync_Seg) is the part of the bit time where edges of the CAN bus level are expected to occur. If an edge occurs outside of Sync_Seg, its distance to the Sync_Seg is called the phase error of this edge.

10.14.1.2 Propagation Time Segment

This part of the bit time is used to compensate physical delay times within the CAN network. These delay times consist of the signal propagation time on the bus and the internal delay time of the CAN nodes.

Any CAN node synchronized to the bit stream on the CAN bus can be out of phase with the transmitter of the bit stream, caused by the signal propagation time between the two nodes. The CAN protocol's non-destructive bitwise arbitration and the dominant acknowledge bit provided by receivers of CAN messages require that a CAN node transmitting a bit stream must also be able to receive dominant bits transmitted by other CAN nodes that are synchronized to that bit stream. The example in figure 10-12 shows the phase shift and propagation times between two CAN nodes.

Figure 10-12. The Propagation Time Segment



Delay A_to_B >= node output delay(A) + bus line delay(AÆB) + node input delay(B)

Prop_Seg >= Delay A_to_B + Delay B_to_A

Prop_Seg >= 2 • [max(node output delay+ bus line delay + node input delay)]

In this example, both nodes A and B are transmitters performing an arbitration for the CAN bus. The node A has sent its Start of Frame bit less than one bit time earlier than node B, therefore node B has synchronized itself to the received edge from recessive to dominant. Since node B has received this edge delay(A_to_B) after it has been transmitted, node B's bit timing segments are shifted with regard to node A. Node B sends an identifier with higher priority and so it will win the arbitration at a specific identifier bit when it transmits a dominant bit while node A transmits a recessive bit. The dominant bit transmitted by node B will arrive at node A after the delay(B_to_A).

Due to oscillator tolerances, the actual position of node A's Sample Point can be anywhere inside the nominal range of node A's Phase Buffer Segments, so the bit transmitted by node B must arrive at node A before the start of Phase_Seg1. This condition defines the length of Prop_Seg.

If the edge from recessive to dominant transmitted by node B would arrive at node A after the start of Phase_Seg1, it could happen that node A samples a recessive bit instead of a dominant bit, resulting in a bit error and the destruction of the current frame by an error flag.

This error only occurs when two nodes arbitrate for the CAN bus which have oscillators of opposite ends of the tolerance range and are separated by a long bus line; this is an example of a minor error in the Bit timing configuration (Prop_Seg too short) that causes sporadic bus errors.

Some CAN implementations provide an optional 3 Sample Mode. The DCAN does not. In this mode, the CAN bus input signal passes a digital low-pass filter, using three samples and a majority logic to determine the valid bit value. This results in an additional input delay of 1 t_q, requiring a longer Prop_Seg.

10.14.1.3 Phase Buffer Segments and Synchronization

The Phase Buffer Segments (Phase_Seg1 and Phase_Seg2) and the Synchronization Jump Width (SJW) are used to compensate for the oscillator tolerance.

The Phase Buffer Segments surround the sample point. The Phase Buffer Segments may be lengthened or shortened by synchronization.

The Synchronization Jump Width (SJW) defines how far the resynchronizing mechanism may move the sample point inside the limits defined by the Phase Buffer Segments to compensate for edge phase errors.

Synchronizations occur on edges from recessive to dominant. Their purpose is to control the distance between edges and sample points.

Edges are detected by sampling the actual bus level in each time quantum and comparing it with the bus level at the previous sample point. A synchronization may be done only if a recessive bit was sampled at the previous sample point and if the actual time quantum's bus level is dominant.

An edge is synchronous if it occurs inside of Sync_Seg, otherwise its distance to the Sync_Seg is the edge phase error, measured in time quanta. If the edge occurs before Sync_Seg, the phase error is negative, else it is positive.

Two types of synchronization exist: hard synchronization and resynchronizing. A hard synchronization is done once at the start of a frame; inside a frame only resynchronization is possible.

- Hard Synchronization
  After a hard synchronization, the bit time is restarted with the end of Sync_Seg, regardless of the edge phase error. Thus hard synchronization forces the edge which has caused the hard synchronization to lie within the synchronization segment of the restarted bit time.
- Bit Resynchronizations
  Resynchronization leads to a shortening or lengthening of the Bit time such that the position of the sample point is shifted with regard to the edge.
  When the phase error of the edge which causes resynchronization is positive, Phase_Seg1 is lengthened. If the magnitude of the phase error is less than SJW, Phase_Seg1 is lengthened by the magnitude of the phase error, else it is lengthened by SJW.
  When the phase error of the edge which causes Resynchronization is negative, Phase_Seg2 is shortened. If the magnitude of the phase error is less than SJW, Phase_Seg2 is shortened by the magnitude of the phase error, else it is shortened by SJW.

If the magnitude of the phase error of the edge is less than or equal to the programmed value of SJW, the results of hard synchronization and resynchronization are the same. If the magnitude of the phase error is larger than SJW, the resynchronization cannot compensate the phase error completely, and an error of (phase error - SJW) remains.

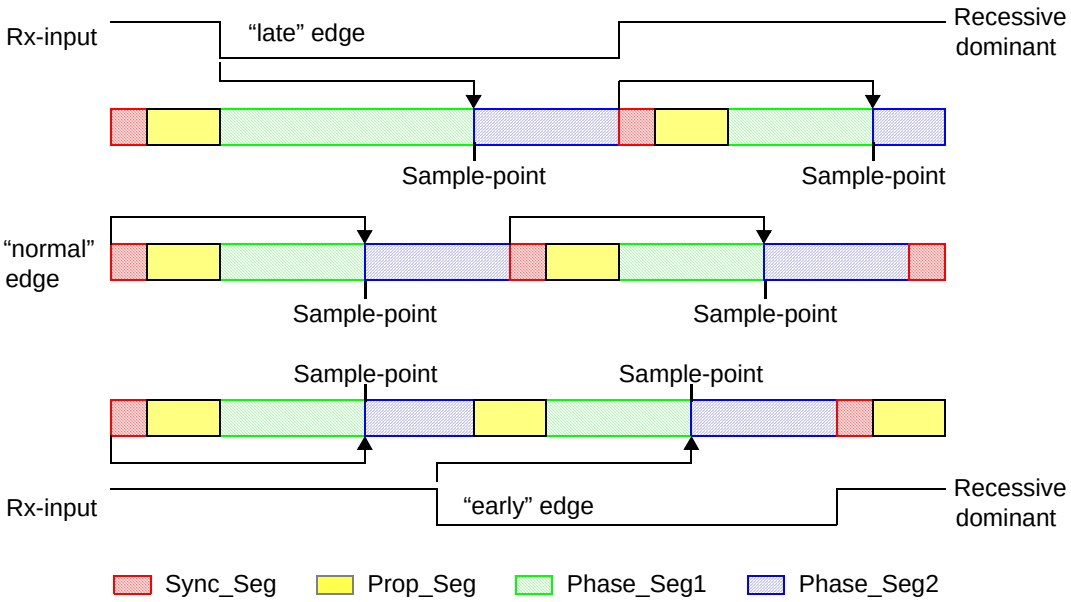
Only one synchronization may be done between two sample points. The synchronizations maintain a minimum distance between edges and sample points, giving the bus level time to stabilize and filtering out spikes that are shorter than (Prop_Seg + Phase_Seg1).

Apart from noise spikes, most synchronizations are caused by arbitration. All nodes synchronize “hard” on the edge transmitted by the “leading” transceiver that started transmitting first, but due to propagation delay times, they cannot become ideally synchronized. The “leading” transmitter does not necessarily win the arbitration, therefore the receivers have to synchronize themselves to different transmitters that subsequently “take the lead” and that are differently synchronized to the previously “leading” transmitter. The same happens at the acknowledge field, where the transmitter and some of the receivers will have to synchronize to that receiver that “takes the lead” in the transmission of the dominant acknowledge bit.

Synchronizations after the end of the arbitration will be caused by oscillator tolerance, when the differences in the oscillator’s clock periods of transmitter and receivers sum up during the time between synchronizations (at most ten bits). These summarized differences may not be longer than the SJW, limiting the oscillator’s tolerance range.

The examples in figure 10-13 show how the Phase Buffer Segments are used to compensate for phase errors. There are three drawings of each two consecutive bit timings. The upper drawing shows the synchronization on a “late” edge, the lower drawing shows the synchronization on an “early” edge, and the middle drawing is the reference without synchronization.

Figure 10-13. Synchronization on Late and Early Edges



In the first example, an edge from recessive to dominant occurs at the end of Prop_Seg. The edge is "late" since it occurs after the Sync_Seg. Reacting to the "late" edge, Phase_Seg1 is lengthened so that the distance from the edge to the sample point is the same as it would have been from the Sync_Seg to the sample point if no edge had occurred. The phase error of this "late" edge is less than SJW, so it is fully compensated and the edge from dominant to recessive at the end of the bit, which is one nominal bit time long, occurs in the Sync_Seg.

In the second example, an edge from recessive to dominant occurs during Phase_Seg2. The edge is "early" since it occurs before a Sync_Seg. Reacting to the "early" edge, Phase_Seg2 is shortened and Sync_Seg is omitted, so that the distance from the edge to the sample point is the same as it would have been from a Sync_Seg to the sample point if no edge had occurred. As in the previous example, the magnitude of this "early" edge's phase error is less than SJW, so it is fully compensated.

The Phase Buffer Segments are lengthened or shortened temporarily only; at the next bit time, the segments return to their nominal programmed values.

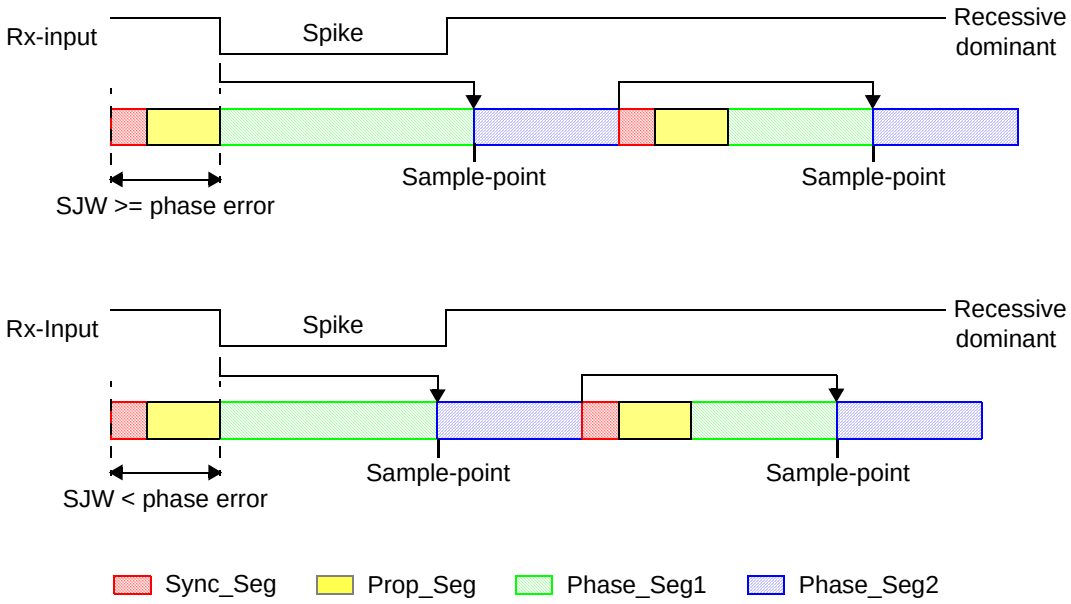
In these examples, the bit timing is seen from the point of view of the CAN implementation's state machine, where the bit time starts and ends at the sample points. The state machine omits Sync_Seg when synchronizing on an "*early*" edge because it cannot subsequently redefine that time quantum of Phase_Seg2 where the edge occurs to be the Sync_Seg.

The examples in Figure 10-14 show how short dominant noise spikes are filtered by synchronizations. In both examples, the spike starts at the end of Prop_Seg and has the length of (Prop_Seg + Phase_Seg1).

In the first example, the Synchronization Jump Width is greater than or equal to the phase error of the spike's edge from recessive to dominant. Therefore the sample point is shifted after the end of the spike; a recessive bus level is sampled.

In the second example, SJW is shorter than the phase error, so the sample point cannot be shifted far enough; the dominant spike is sampled as actual bus level.

Figure 10-14. Filtering of Short Dominant Spikes



10.14.1.4 Oscillator Tolerance Range

With the introduction of CAN protocol version 1.2, the option to synchronize on edges from dominant to recessive became obsolete. Only edges from recessive to dominant are considered for synchronization. The protocol update to version 2.0 (A and B) had no influence on the oscillator tolerance.

The tolerance range df for an oscillator's frequency f_{osc} around the nominal frequency f_{nom} with

$$(1 - df) \bullet f_{nom} \leq f_{osc} \leq (1 + df) \bullet f_{nom}$$

depends on the proportions of Phase_Seg1, Phase_Seg2, SJW, and the bit time. The maximum tolerance df is the defined by two conditions (both shall be met):

$$\text{I:} \quad df \leq \frac{\min(TSeg1, TSeg2)}{2x(13x(bit_time - TSeg2))}$$

$$\text{II:} \quad df \leq \frac{SJW}{20x bit_time}$$

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.

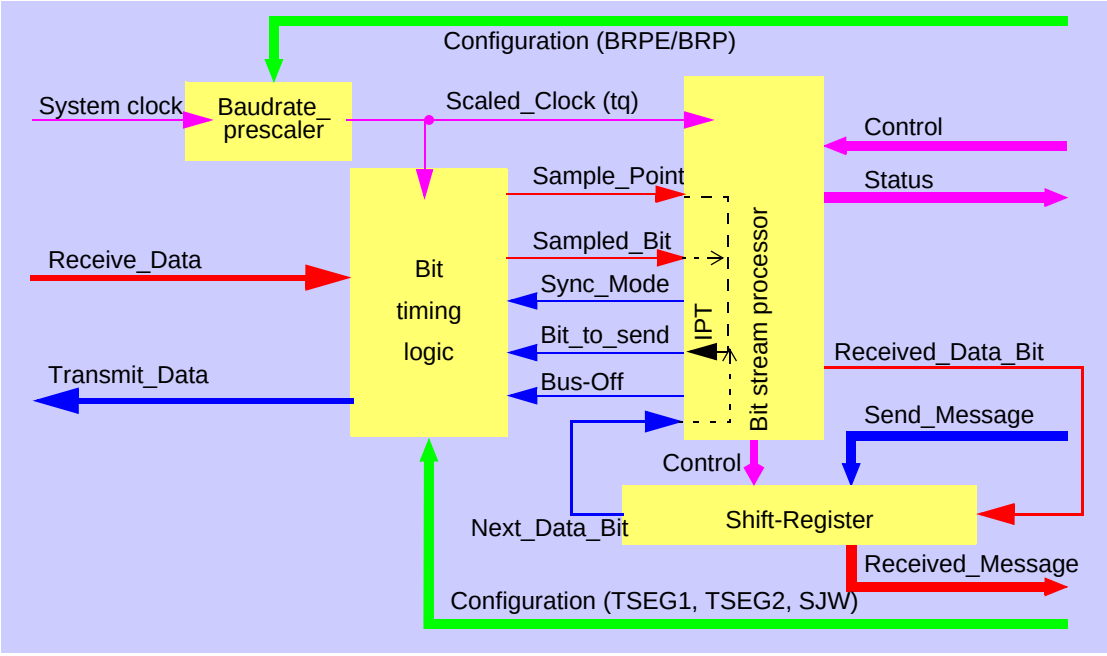
The combination Prop_Seg = 1 and Phase_Seg1 = Phase_Seg2 = SJW = 4 allows the largest possible oscillator tolerance of 1.58%. This combination with a Propagation Time Segment of only 10% of the bit time is not suitable for short bit times; it can be used for bit rates of up to 125 kBit/s (bit time = 8 µs) with a bus length of 40 m.

10.14.2 Configuration of the DCAN Bit Timing

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 (BRPE) is provided. The sum of Prop_Seg and Phase_Seg1 (as TSEG1) is combined with Phase_Seg2 (as TSEG2) in one byte, SJW and BRP (plus BRPE in third byte) are combined in the other byte (see Figure 10-15)

.

Figure 10-15. Structure of the CAN Core's CAN Protocol Controller



In this bit timing register, the components TSEG1, TSEG2, SJW and BRP have to be programmed to a numerical value that is one less than its functional value; so instead of values in the range of [1…n], values in the range of [0…n-1] are programmed. That way, e.g. SJW (functional range of [1…4]) is represented by only two bits.

Therefore the length of the Bit time is (programmed values) [TSEG1 + TSEG2 + 3] t_q or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] t_q.

The data in the Bit Timing Register is the configuration input of the CAN protocol controller. The Baud Rate Prescaler (configured by BRPE/BRP) defines the length of the time quantum (the basic time unit of the bit time); the Bit Timing Logic (configured by TSEG1, TSEG2, and SJW) defines the number of time quanta in the bit time.

The processing of the bit time, the calculation of the position of the Sample Point, and occasional synchronizations are controlled by the Bit timing state machine, which is evaluated once each time quantum. The rest of the CAN protocol controller, the Bit Stream Processor (BSP) state machine, is evaluated once each bit time, at the Sample Point.

The Shift Register serializes the messages to be sent and parallelizes received messages. Its loading and shifting is controlled by the BSP.

The BSP translates messages into frames and vice versa. It generates and discards the enclosing fixed format bits, inserts and extracts stuff bits, calculates and checks the CRC code, performs the error management, and decides which type of synchronization is to be used. It is evaluated at the sample point and processes the sampled bus input bit. The time after the sample point that is needed to calculate the next bit to be sent (e.g. data bit, CRC bit, stuff bit, error flag, or idle) is called the Information Processing Time (IPT), which is 0 t_q for the DCAN.

Generally, the IPT is CAN controller specific, but may not be longer than 2 t_q. The IPC length is the lower limit of the programmed length of Phase_Seg2. In case of a synchronization, Phase_Seg2 may be shortened to a value less than IPT, which does not affect bus timing.

10.14.2.1 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.

The oscillator tolerance range necessary for the resulting configuration is calculated by the formulas given in section 10.14.1.4

If more than one configurations are possible to reach a certain Bit rate, it is recommended to choose the configuration which allows the highest oscillator tolerance range.

CAN nodes with different clocks require different configurations to come to the same bit rate. The calculation of the propagation time in the CAN network, based on the nodes with the longest delay times, is done once for the whole network.

The CAN system's oscillator tolerance range is limited by the node with the lowest tolerance range.

The calculation may show that bus length or bit rate have to be decreased or that the oscillator frequencies' stability has to be increased in order to find a protocol compliant configuration of the CAN bit timing.

The resulting configuration is written into the Bit Timing Register:

```
(Phase_Seg2-1)&(Phase_Seg1+Prop_Seg-1)&
(SynchronizationJumpWidth-1)&(Prescaler-1)
```

10.14.2.2 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	90	ns		
delay of receiver circuit	40	ns		
delay of bus line (40m)	220	ns		
t_{Prop}	700	ns	=	2*delays = 7 • t_q
t_{SJW}	100	ns	=	1 • t_q
t_{TSeg1}	800	ns	=	t_{Prop} + t_{SJW}
t_{TSeg2}	100	ns	=	Information Processing Time + 1 • t_q
$t_{Sync\text{-}Seg}$	100	ns	=	1 • t_q
bit time	1000	ns	=	$t_{Sync\text{-}Seg}$ + t_{TSeg1} + t_{TSeg2}
tolerance for CAN_CLK	0.43	%	=	$\frac{\min(TSeg1, TSeg2)}{2x(13x(bit_time - TSeg2))}$
			=	$\frac{0.1\mu s}{2x(13x(1\mu s - 0.1\mu s))}$

In this example, the concatenated bit time parameters are $(1\text{-}1)_3\&(8\text{-}1)_4\&(1\text{-}1)_2\&(1\text{-}1)_6$, so the Bit Timing Register is programmed to= 0x00000700.

10.14.2.3 Example for Bit Timing at low Baudrate

In this example, the frequency of CAN_CLK is 2 MHz, BRP is 1, the bit rate is 100 KBit/s.

t_q	1	µs	=	2 • t_{CAN_CLK}
delay of bus driver	200	ns		
delay of receiver circuit	80	ns		
delay of bus line (40m)	220	ns		
t_{Prop}	1	µs	=	1 • t_q
t_{SJW}	4	µs	=	4 • t_q
t_{TSeg1}	5	µs	=	t_{Prop} + t_{SJW}
t_{TSeg2}	4	µs	=	Information Processing Time + 4 • t_q
$t_{Sync\text{-}Seg}$	1	µs	=	1 • t_q
bit time	10	µs	=	$t_{Sync\text{-}Seg}$ + t_{TSeg1} + t_{TSeg2}
tolerance for CAN_CLK	3.08	%	=	$\frac{\min(TSeg1, TSeg2)}{2x(13x(bit_time - TSeg2))}$
			=	$\frac{4\mu s}{2x(13x(9\mu s - 4\mu s))}$

In this example, the concatenated bit time parameters are $(4\text{-}1)_3\&(5\text{-}1)_4\&(4\text{-}1)_2\&(2\text{-}1)_6$, so the Bit Timing Register is programmed to= 0x000034C1.

10.15 Message Interface Register Sets

The Interface Register sets control the CPU read and write accesses to the Message RAM. There are two Interface Registers Sets for read / write access (IF1 and IF2) and one Interface Register Set for read access only (IF3).

Due to the structure of the Message RAM, it is not possible to change single bits or bytes of a message object. Instead, always a complete message object in the Message RAM is accessed. Therefore the data transfer from the IF1/IF2 Registers to the Message RAM requires the Message Handler to perform a read-modify-write cycle: First those parts of the message object that are not to be changed are read from the Message RAM into the Interface Register set, and after the update the whole content of the Interface Register set is written into the message object.

After the partial write of a message object, those parts of the Interface Register set which are not selected in the Command Register, will be set to the actual contents of the selected message object.

After the partial read of a message object, those parts of the Interface Register set which are not selected in the Command Register, will be left unchanged.

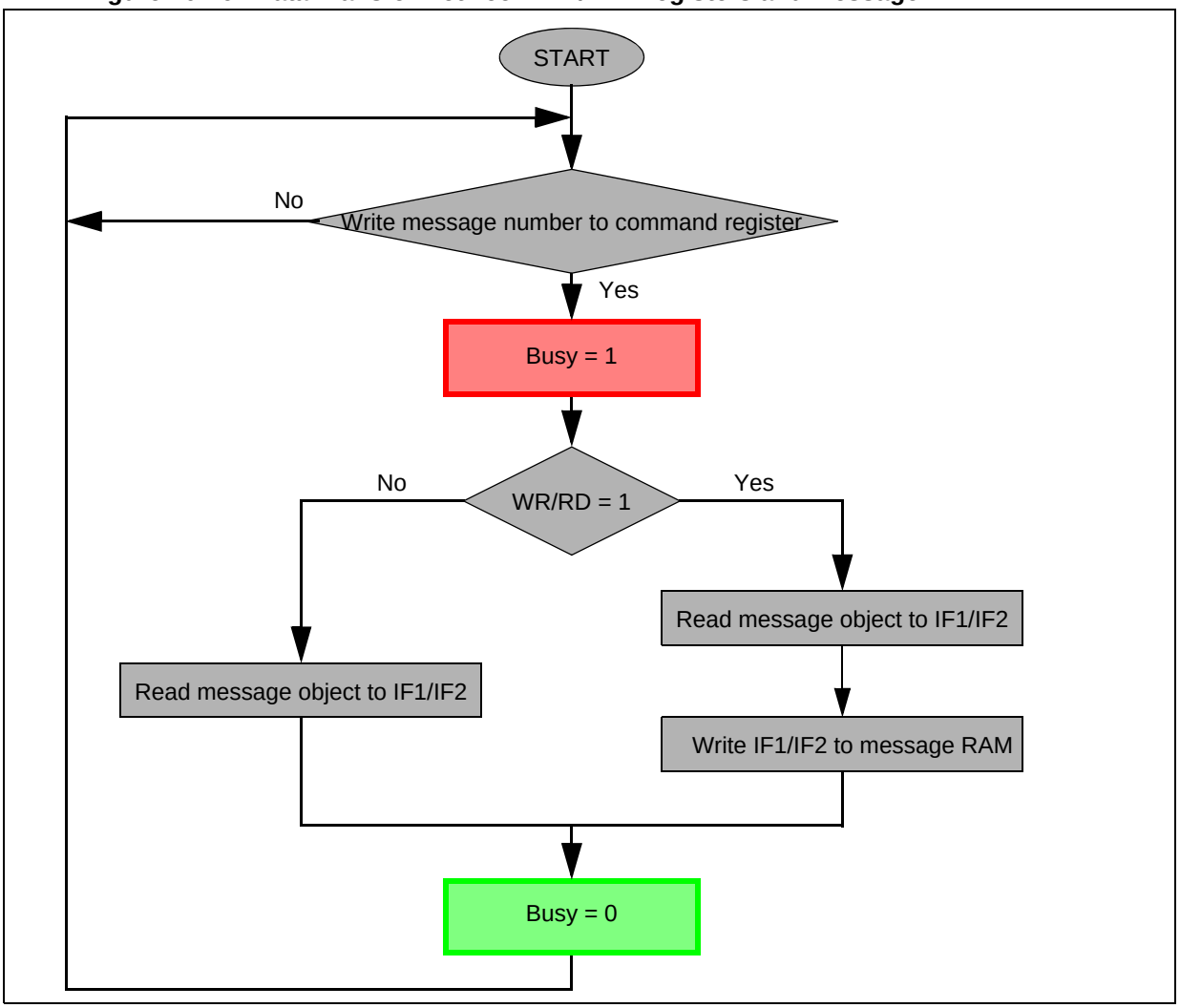
By buffering the data to be transferred, the Interface Register sets avoid conflicts between concurrent CPU accesses to the Message RAM and CAN message reception and transmission. A complete message object (see section 10.16.1) or parts of the message object may be transferred between the Message RAM and the IF1/IF2 Register set (see section 10.17.20) in one single transfer. This transfer, performed in parallel on all selected parts of the message object, guarantees the data consistency of the CAN message.

10.15.1 Message Interface Register Sets 1 and 2

The IF1 and IF2 Registers Sets control the data transfer to and from the message object. The Command Register addresses the desired message object in the Message RAM and specifies whether a complete message object or only parts should be transferred. The data transfer is initiated by writing the message number to the bits [7:0] of the Command Register.

When the CPU initiates a data transfer between the IF1/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 10-16).

Figure 10-16. Data Transfer Between IF1 / IF2 Registers and Message RAM



10.15.2 IF3 Register Set

The IF3 register set can automatically be updated with received message objects without the need to initiate the transfer from Message RAM by CPU. The automatic update functionality can be programmed for each message object (see IF3 Update Enable register, section 10.17.29).

All valid message objects in Message RAM which are configured for automatic update, will be checked for active NewDat flags. If such a message object is found, it will be transferred to the IF3 register (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, an IF3 interrupt can also be generated.

Note:
The IF3 register set can not be used for transferring data into message objects.

10.16 Message RAM

The DCAN Message RAM contains message objects and parity bits for the message objects. There are up to 128 message objects in the Message RAM.

During normal operation, accesses to the Message RAM are performed via the Interface Register sets, and the CPU cannot directly access the Message RAM.

The Interface Register sets IF1 and IF2 provide indirect read/write access from the CPU to the Message RAM. The IF1 and IF2 register sets can buffer control and user data to be transferred to and from the message objects.

The third Interface Register set IF3 can be configured to automatically receive control and user data from the Message RAM when a message object has been updated after reception of a CAN message. The CPU does not need to initiate the transfer from Message RAM to 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 10.16.3)
2. In RAM Direct Access (RDA) mode (see section 10.16.4)

For the Message RAM Base address, please refer to the device datasheet.

10.16.1 Structure of Message Objects

Figure 10-17 shows the structure of a message object.

The grayed fields are those parts of the message object which are represented in dedicated registers. For example, the transmit request flags of all message objects are represented in centralized transmit request registers.

Figure 10-17. Structure of a Message Object

Message Object												
UMask	Msk[28:0]	MXtd	MDir	EoB	unused	NewDat	MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
MsgVal	ID[28:0]	Xtd	Dir	DLC[3:0]	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7

.

Table 10-2. Message Object Field Descriptions

Name	Value	Description
MsgVal		Message valid
	0	The message object is ignored by the Message Handler.
	1	The message object is to be used by the Message Handler.
		Note: This bit may be kept at level ‘1’ even when the identifier bits ID[28:0], the control bits Xtd, Dir, or the Data Length Code DLC[3:0] are changed. It should be reset if the Messages Object is no longer required.
UMask		Use Acceptance Mask
	0	Mask bits (Msk[28:0], MXtd and MDir) are ignored and not used for acceptance filtering.
	1	Mask bits are used for acceptance filtering.
		Note: If the UMask bit is set to one, the message object’s mask bits have to be programmed during initialization of the message object before MsgVal is set to one.
ID[28:0]		Message Identifier
	ID[28:0]	29-bit (“extended”) identifier bits
	ID[28:18]	11-bit (“standard”) identifier bits
Msk[28:0]		Identifier Mask
	0	The corresponding bit in the message identifier is not used for acceptance filtering (don’t care).
	1	The corresponding bit in the message identifier is used for acceptance filtering.
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

Table 10-2. Message Object Field Descriptions

Name	Value	Description
	0	The extended identifier bit (IDE) has no effect on the acceptance filtering.
	1	The extended identifier bit (IDE) is used for acceptance filtering. Note: When 11-bit (“standard”) Identifiers are used for a message object, the identifiers of received data frames are written into bits ID[28:18]. For acceptance filtering, only these bits together with mask bits Msk[28:18] are considered.
Dir		Message Direction
	0	Direction = receive: On TxRqst, a remote frame with the identifier of this message object is transmitted. On reception of a data frame with matching identifier, the message is stored in this message object.
	1	Direction = transmit: On TxRqst, a data frame is transmitted. On reception of a remote frame with matching identifier, the TxRqst bit of this message object is set (if RmtEn = one).
MDir		Mask Message Direction
	0	The message direction bit (Dir) has no effect on the acceptance filtering.
	1	The message direction bit (Dir) is used for acceptance filtering.
EOB		End of Block
	0	The message object is part of a FIFO Buffer block and is not the last message object of this FIFO Buffer block.
	1	The message object is a single message object or the last message object in a FIFO Buffer Block. Note: This bit is used to concatenate multiple message objects to build a FIFO Buffer. For single message objects (not belonging to a FIFO Buffer), this bit must always be set to one.
NewDat		New Data
	0	No new data has been written into the data bytes of this message object by the Message Handler since the last time when this flag was cleared by the CPU.
	1	The Message Handler or the CPU has written new data into the data bytes of this message object.
MsgLst		Message Lost (only valid for Message Objects with direction = receive)
	0	No message was lost since the last time when this bit was reset by the CPU.
	1	The Message Handler stored a new message into this message object when NewDat was still set, so the previous message has been overwritten.
RxIE		Receive Interrupt Enable
	0	IntPnd will not be triggered after the successful reception of a frame.
	1	IntPnd will be triggered after the successful reception of a frame.

Table 10-2. Message Object Field Descriptions

Name	Value	Description
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.
		Note: See section 10.13.8 for details on the setup of RmtEn and UMask for remote frames.
TxRqst		Transmit Request
	0	This message object is not waiting for a transmission.
	1	The transmission of this message object is requested and is not yet done.
DLC[3:0]		Data Length Code
	0-8	Data Frame has 0-8 data bits.
	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.

Table 10-2. Message Object Field Descriptions

Name	Value	Description
Data 0		1st data byte of a CAN Data Frame
Data 1		2nd data byte of a CAN Data Frame
Data 2		3rd data byte of a CAN Data Frame
Data 3		4th data byte of a CAN Data Frame
Data 4		5th data byte of a CAN Data Frame
Data 5		6th data byte of a CAN Data Frame
Data 6		7th data byte of a CAN Data Frame
Data 7		8th data byte of a CAN Data Frame Note: Byte Data 0 is the first data byte shifted into the shift register of the CAN Core during a reception, byte Data 7 is the last. When the Message Handler stores a data frame, it will write all the eight data bytes into a message object. If the Data Length Code is less than 8, the remaining bytes of the message object may be overwritten by undefined values.

10.16.2 Addressing Message Objects in RAM

The starting location of a particular message object in RAM is:

Message RAM base address + (message object number) * 0x20.

This means that Message Object 1 starts at offset 0x0020; Message Object 2 starts at offset 0x0040, etc.

Note:
'0' is not a valid message object number.

Note:
Do not write to mailboxes beyond the number described in the device datasheet. A write to a mailbox address beyond the implemented number of mailboxes might corrupt an implemented mailbox.

Message Object number 1 has the highest priority.

Table 10-3. Message RAM addressing in Debug/Suspend and RDA mode

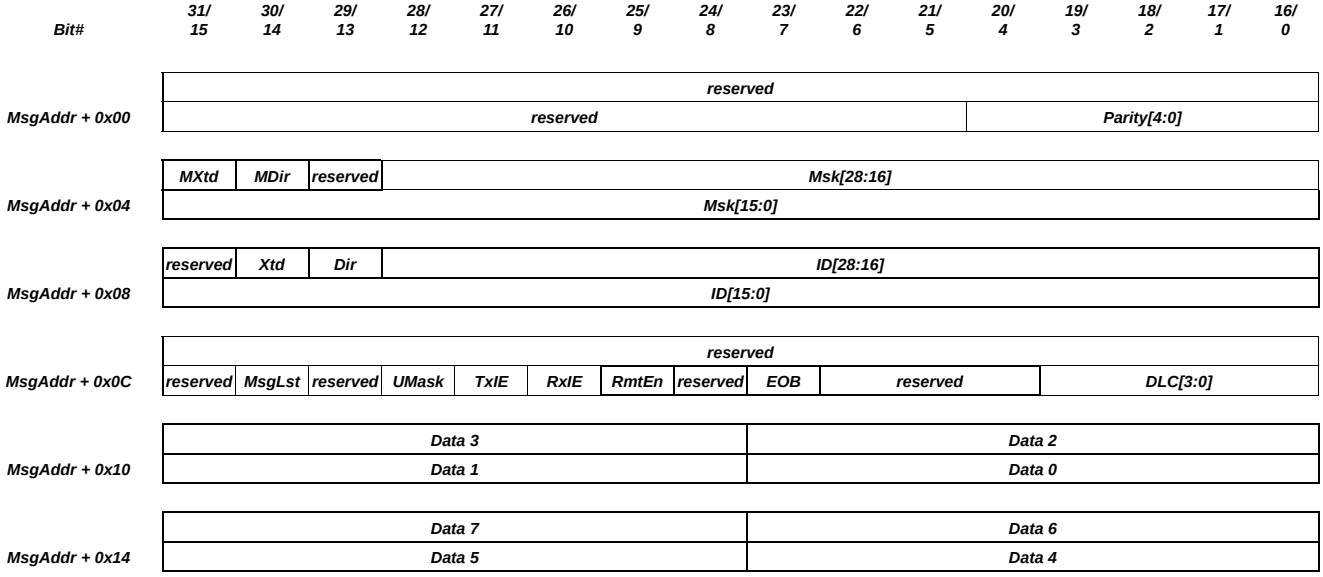
Message Object number	offset from base address	word number	Debug/Suspend mode, see section 10.16.3	RDA mode, see section 10.16.4
last implemented (here:128)	**0x0000**	1	Parity	Data Bytes 4-7
	0x0004	2	MXtd,MDir,Mask	Data Bytes 0-3
	0x0008	3	Xtd,Dir,ID	ID[27:0],DLC
	0x000C	4	Ctrl	Mask,Xtd,Dir,ID[28]
	0x0010	5	Data Bytes 3-0	Parity,Ctrl,MXtd,MDir
	0x0014	6	Data Bytes 7-4	--
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	--
2	**0x0040**	1	Parity	Data Bytes 4-7
	0x0044	2	MXtd,MDir,Mask	Data Bytes 0-3
	0x0048	3	Xtd,Dir,ID	ID[27]:0,DLC
	0x004C	4	Ctrl	Mask,Xtd,Dir,ID[28]
	0x0050	5	Data Bytes 3-0	Parity,Ctrl,MXtd,MDir
	0x0054	6	Data Bytes 7-4	--
..	...	...	...	...
127	**0x0FE0**	1	Parity	Data Bytes 4-7
	0x0FE4	2	MXtd,MDir,Mask	Data Bytes 0-3
	0x0FE8	3	Xtd,Dir,ID	ID[27:0],DLC
	0x0FEC	4	Ctrl	Mask,Xtd,Dir,ID[28]
	0x0FF0	5	Data Bytes 3-0	Parity,Ctrl,MXtd,MDir
	0x0FF4	6	Data Bytes 7-4	--

10.16.3 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 10-18. Message RAM representation in Debug/Suspend Mode



10.16.4 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 splitted 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.

Any read or write to the RAM addresses for RamDirectAccess during normal operation mode (TestMode bit or RDA bit not set) will be ignored.

Figure 10-19. Message RAM representation in RAM Direct Access Mode

Bit#	31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0
MsgAddr + 0x00	Data 4								Data 5							
	Data 6								Data 7							
MsgAddr + 0x04	Data 0								Data 1							
	Data 2								Data 3							
MsgAddr + 0x08	ID[27:12]															
	ID[11:0]												DLC[3:0]			
MsgAddr + 0x0C	Msk[28:13]															
	Msk[12:0]													Xtd	Dir	ID[28]
MsgAddr + 0x10	reserved															Parity[4]
	Parity[3:0]				unused				MsgLst	UMask	TxIE	RxIE	RmtEn	EOB	MXtd	MDir

Note:
'unused' means here that those bits can be written to but have no effect.

10.17 DCAN Control Registers

Figure 10-20. DCAN Control 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
0x00 DCAN CTL	Reserved						WUBA	PDR	Reserved			DE3	DE2	DE1	IE1	InitDbg
	SWR	Reserved	PMD				ABO	IDS	Test	CCE	DAR	Reserved	EIE	SIE	IE0	Init
0x04 DCAN ES	Reserved															
	Reserved					PDA	Wake UpPnd	PER	BOff	EWarn	EPass	RxOK	TxOK	LEC		
0x08 DCAN ERRC	Reserved															
	RP	REC[6:0]							TEC[7:0]							
0x0C DCAN BTR	Reserved												BRPE			
	Reserved	TSeg2			TSeg1				SJW		BRP					
0x10 DCAN INT	Reserved								Int1ID[7:0]							
	Int0ID[15:0]															
0x14 DCAN TEST	Reserved															
	Reserved						RDA	EXL	Rx	Tx[1:0]		LBack	Silent	Reserved		
0x1C DCAN PERR	Reserved															
	Reserved					Word Number			Message Number							
0x20 DCAN REL	REL				STEP				SUBSTEP				YEAR			
	MON								DAY							

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
0x80 DCAN ABOTR	ABO Time[31:16]															
	ABO Time[15:0]															
0x84 DCAN TXRQ X	Reserved															
	TxRqstReg8		TxRqstReg7		TxRqstReg6		TxRqstReg5		TxRqstReg4		TxRqstReg3		TxRqstReg2		TxRqstReg1	
0x88 DCAN TXRQ12	TxRqst[32:17]															
	TxRqst[16:1]															
0x8C DCAN TXRQ34	TxRqst[64:49]															
	TxRqst[48:33]															
0x90 DCAN TXRQ56	TxRqst[96:81]															
	TxRqst[80:65]															
0x94 DCAN TXRQ78	TxRqst[128:113]															
	TxRqst[112:97]															
0x98 DCAN NWDAT X	Reserved															
	NewDatReg8		NewDatReg7		NewDatReg6		NewDatReg5		NewDatReg4		NewDatReg3		NewDatReg2		NewDatReg1	
0x9C DCAN NWDAT12	NewDat[32:17]															
	NewDat[16:1]															
0xA0 DCAN NWDAT34	NewDat[64:49]															
	NewDat[48:33]															

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
0xA4 DCAN NWDAT56	NewDat[96:81]															
	NewDat[80:65]															
0xA8 DCAN NWDAT78	NewDat[128:113]															
	NewDat[112:97]															
0xAC DCAN INTPND X	Reserved															
	IntPndReg8				IntPndReg7				IntPndReg6				IntPndReg5			
	IntPndReg4				IntPndReg3				IntPndReg2				IntPndReg1			
0xB0 DCAN INTPND12	IntPnd[32:17]															
	IntPnd[16:1]															
0xB4 DCAN INTPND34	IntPnd[64:49]															
	IntPnd[48:33]															
0xB8 DCAN INTPND56	IntPnd[96:81]															
	IntPnd[80:65]															
0xBC DCAN INTPND78	IntPnd[128:113]															
	IntPnd[112:97]															
0xC0 DCAN MSGVAL X	Reserved															
	MsgValReg8				MsgValReg7				MsgValReg6				MsgValReg5			
	MsgValReg4				MsgValReg3				MsgValReg2				MsgValReg1			
0xC4 DCAN MSGVAL12	MsgVal[32:17]															
	MsgVal[16:1]															

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
0xC8	MsgVal[64:49]															
DCAN MSGVAL34	MsgVal[48:33]															
0xCC	MsgVal[96:81]															
DCAN MSGVAL56	MsgVal[80:65]															
0xD0	MsgVal[128:113]															
DCAN MSGVAL78	MsgVal[112:97]															
0xD8	IntMux[32:17]															
DCAN INTMUX12	IntMux[16:1]															
0xDC	IntMux[64:49]															
DCAN INTMUX34	IntMux[48:33]															
0xE0	IntMux[96:81]															
DCAN INTMUX56	IntMux[80:65]															
0xE4	IntMux[128:113]															
DCAN INTMUX78	IntMux[112:97]															
0x100	Reserved									WR/RD	Mask	Arb	Control	Clr IntPnd	TxRqst/ New Dat	Data A	Data B
DCAN IF1CMD	Busy	DMA active	Reserved							Message Number							
0x104	MXtd	MDir	Reserv ed	Msk[28:16]													
DCAN IF1MSK	Msk[15:0]																

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
0x108 DCAN IF1ARB	MsgVal	Xtd	Dir	ID[28:16]												
	ID[15:0]															
0x10C DCAN IF1MCTL	Reserved															
	New Dat	Msg Lst	Int Pnd	UMask	TxIE	RxIE	Rmt En	Tx Rqst	EoB	Reserved			DLC[3:0]			
0x110 DCAN IF1DATA	Data 3								Data 2							
	Data 1								Data 0							
0x114 DCAN IF1DATB	Data 7								Data 6							
	Data 5								Data 4							
0x120 DCAN IF2CMD	Reserved								WR/RD	Mask	Arb	Control	Clr IntPnd	TxRqst/ New-Dat	Data A	Data B
	Busy	DMA active	Reserved						Message Number							
0x124 DCAN IF2MSK	MXtd	MDir	Reserved	Msk[28:16]												
	Msk[15:0]															
0x128 DCAN IF2ARB	MsgVal	Xtd	Dir	ID[28:16]												
	ID[15:0]															
0x12C DCAN IF2MCTL	Reserved															
	New Dat	Msg Lst	Int Pnd	UMask	TxIE	RxIE	Rmt En	Tx Rqst	EoB	Reserved			DLC[3:0]			
0x130 DCAN IF2DATA	Data 3								Data 2							
	Data 1								Data 0							

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
0x134 DCAN IF2DATB	Data 7								Data 6							
	Data 5								Data 4							
0x140 DCAN IF3OBS	Reserved															
	IF3 Upd	Reserved		IF3 SDB	IF3 SDA	IF3 SC	IF3 SA	IF3 SM	Reserved			Data B	DataA	Ctrl	Arb	Mask
0x144 DCAN IF3MSK	MXtd	MDir	Reserved	Msk[28:16]												
	Msk[15:0]															
0x148 DCAN IF3ARB	MsgVal	Xtd	Dir	ID[28:16]												
	ID[15:0]															
0x14C DCAN IF3MCTL	Reserved															
	New Dat	Msg Lst	Int Pnd	UMask	TxIE	RxIE	Rmt En	Tx Rqst	EoB	Reserved		DLC[3:0]				
0x150 DCAN IF3DATA	Data 3								Data 2							
	Data 1								Data 0							
0x154 DCAN IF3DATB	Data 7								Data 6							
	Data 5								Data 4							
0x160 DCAN IF3UPD12	IF3UpdEn[32:17]															
	IF3UpdEn[16:1]															
0x164 DCAN IF3UPD34	IF3UpdEn[64:49]															
	IF3UpdEn[48:33]															

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
0x168 DCAN IF3UPD56	IF3UpdEn[96:81]															
	IF3UpdEn[80:65]															

0x16C DCAN IF3UPD78	IF3UpdEn[128:113]															
	IF3UpdEn[112:97]															

0x1E0 DCAN TIOC	Reserved													PU	PD	OD
	Reserved												Func	Dir	Out	In

0x1E4 DCAN RIOC	Reserved													PU	PD	OD
	Reserved												Func	Dir	Out	In

After hardware reset, the registers of the DCAN hold the values shown in the register descriptions.

Additionally, the Bus-Off state is reset and the CAN_TX pin is set to recessive (HIGH). The Init bit in the CAN Control Register is set to enable the software initialization. The DCAN will not influence the CAN bus until the CPU resets Init to ‘0’.

10.17.1 CAN Control Register (DCAN CTL)

Figure 10-21. CAN Control Register (DCAN CTL) [offset = 0x00]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						WUBA	PDR	Reserved			DE3	DE2	DE1	IE1	InitDbg
R-0						R/W-0	R/W-0	R-0			R/W-0	R/W-0	R/W-0	R/W-0	R-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWR	Reserved	PMD				ABO	IDS	Test	CCE	DAR	Reserved	EIE	SIE	IE0	Init
R/WP-0	R-0	R/W-0x5				R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-1

R = Read, W = Write, WP = Write Protected by Init bit, *-n* = Value after reset

Table 10-4. CAN Control Register Field Descriptions

Bit	Name	Value	Description
31-26	Reserved		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 (see also section 10.8).
			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 (see also section 10.8).
23-21	Reserved		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.

Table 10-4. CAN Control Register Field Descriptions (Continued)

Bit	Name	Value	Description
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 one; line remains active until pending interrupts are processed.
16	InitDbg		Internal init state while debug access
		0	Not in debug mode, or debug mode requested but not entered.
		1	Debug mode requested and internally entered; the 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		This bit is always read as 0. Writes have no effect.
13-10	PMD		Parity on/off
		0x5	Parity function disabled
		Others	Parity function enabled
9	ABO		Auto-Bus-On Enable
		0	The Auto-Bus-On feature is disabled
		1	The Auto-Bus-On feature is enabled

Table 10-4. CAN Control Register Field Descriptions (Continued)

Bit	Name	Value	Description
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 configuration registers.
		1	The CPU has write access to the configuration registers (when Init bit is set).
5	DAR		Disable Automatic Retransmission
		0	Automatic Retransmission of not successful messages enabled.
		1	Automatic Retransmission disabled.
4	Reserved		This bit is always read as 0. Writes have no effect.
3	EIE		Error Interrupt Enable
		0	Disabled - PER, BOff and EWarn bits can not 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 can not 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 one; 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 Rev. 2.0) cannot be shortened by setting or resetting Init bit. If the module goes Bus-Off, it will automatically set the Init bit and stop all bus activities.

When the Init bit is cleared by the application again, the module will then wait for 129 occurrences of Bus Idle (129 * 11 consecutive recessive bits) before resuming normal operation. At the end of the Bus-Off recovery sequence, the error counters will be reset.

After the Init bit is reset, each time when a sequence of 11 recessive bits is monitored, a Bit0 Error code is written to the Error and Status Register, enabling the CPU to check whether the CAN bus is stuck at dominant or continuously disturbed, and to monitor the proceeding of the Bus-Off recovery sequence.

10.17.2 Error and Status Register (DCAN ES)

Figure 10-22. Error and Status Register (DCAN ES) [offset = 0x04]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					PDA	Wake UpPnd	PER	BOff	EWarn	EPass	RxOK	TxOK	LEC		
R-0					R-0	R/C-0	R/C-0	R-0	R-0	R-0	R/C-0	R/C-0	R/S-111		

R = Read, S = Set by Read, C = Clear by Read, *-n* = Value after reset

.

Table 10-5. Error and Status Register Field Descriptions

Bit	Name	Value	Description
31–11	Reserved		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.

Table 10-5. Error and Status Register Field Descriptions

Bit	Name	Value	Description
6	EWarn		Warning State
		0	Both error counters are below the error warning limit of 96.
		1	At least one of the error counters has reached the error warning limit of 96.
5	EPass		Error Passive State
		0	On CAN Bus error, the 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.

Table 10-5. Error and Status Register Field Descriptions

Bit	Name	Value	Description
2-0	LEC		Last Error Code The LEC field indicates the type of the last error on the CAN bus. This field will be cleared to '0' when a message has been transferred (reception or transmission) without error.
		0	No Error
		1	Stuff Error**:** More than five equal bits in a row have been detected in a part of a received message where this is not allowed.
		2	Form Error**:** A fixed format part of a received frame has the wrong format.
		3	Ack Error**:** The message this CAN Core transmitted was not acknowledged by another node.
		4	Bit1 Error**:** During the transmission of a message (with the 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.
		5	Bit0 Error**:** During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (logical value '0'), but the monitored bus level was recessive. During Bus-Off recovery, this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the Bus-Off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed).
		6	CRC Error: In a received message, the CRC check sum was incorrect. (CRC received for an incoming message does not match the calculated CRC for the received data).
		7	No CAN bus event was detected since the last time when CPU has read the Error and Status Register. Any read access to the Error and Status Register re-initializes the LEC to value '7'.

Interrupts are generated by bits PER, BOff and EWarn (if EIE bit in CAN Control Register is set) and by bits WakeUpPnd, RxOk, TxOk, and LEC (if SIE bit in CAN Control Register is set).

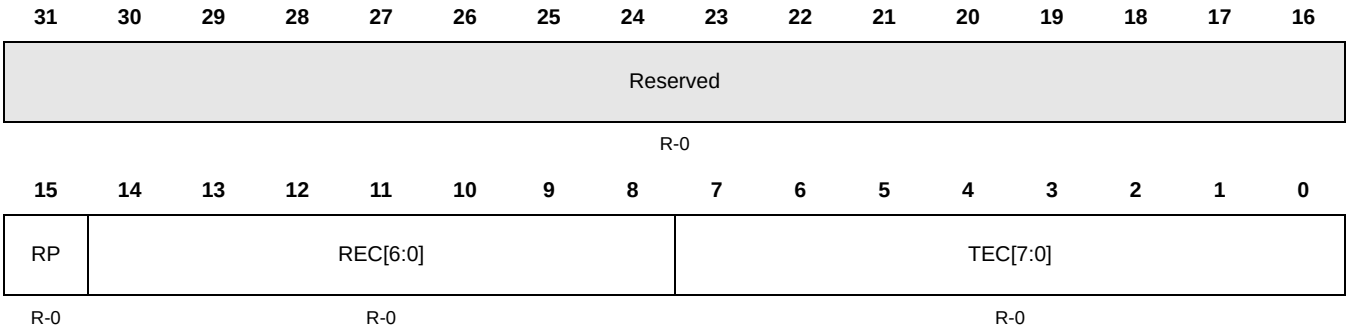
A change of bit EPass will not generate an Interrupt.

Note:
Reading the Error and Status Register clears the WakeUpPnd, PER, RxOk and TxOk bits and set the LEC to value '7'. Additionally, the Status Interrupt value (0x8000) in the Interrupt Register will be replaced by the next lower priority interrupt value.

Note:
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.

10.17.3 Error Counter Register (DCAN ERRC)

Figure 10-23. Error Counter Register (DCAN ERRC) [offset = 0x08]



R = Read, *-n* = Value after reset

.

Table 10-6. Error Counter Register Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		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[6:0]		Receive Error Counter. Actual state of the Receive Error Counter. (values from 0 to 255).
7-0	TEC[7:0]		Transmit Error Counter. Actual state of the Transmit Error Counter. (values from 0 to 255).

10.17.4 Bit Timing Register (DCAN BTR)

Figure 10-24. Bit Timing Register (DCAN BTR) [offset = 0x0C]



R = Read, WP = Write Protected by CCE bit, *-n* = Value after reset

.

Table 10-7. Bit Timing Register Field Descriptions

Bit	Name	Value	Description
31–20	Reserved		These bits are always read as 0. Writes have no effect.
19-16	BRPE	0x00-0x0F	Baud Rate Prescaler Extension. Valid programmed values are 0 to 15. By programming BRPE the Baud Rate Prescaler can be extended to values up to 1024.
15	Reserved		This bit is always read as 0. Writes have no effect.
14-12	TSeg2	0x0-0x7	Time segment after the sample point Valid programmed values are 0 to 7. The actual TSeg2 value which is interpreted for the Bit Timing will be the programmed TSeg2 value + 1.
11-8	TSeg1	0x01-0x0F	Time segment before the sample point Valid programmed values are 1 to15. The actual TSeg1 value interpreted for the Bit Timing will be the programmed TSeg1 value + 1.
7-6	SJW	0x0-0x3	Synchronization Jump Width Valid programmed values are 0 to 3. The actual SJW value interpreted for the Synchronization will be the programmed SJW value + 1.
5-0	BRP	0x00-0x3F	Baud Rate Prescaler Value by which the CAN_CLK frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid programmed values are 0 to 63. The actual BRP value interpreted for the Bit Timing will be the programmed BRP value + 1.

Note:

This register is only writable if CCE and Init bits in the CAN Control Register are set.

Note:
The CAN bit time may be programmed in the range of 8 to 25 time quanta.

Note:
The CAN time quantum may be programmed in the range of 1 to1024 CAN_CLK periods.

With a CAN_CLK of 8 MHz and BRPE = 0x00, the reset value of 0x00002301 configures the DCAN for a bit rate of 500kBit/s.

For details see section 10.14.

10.17.5 Interrupt Register (DCAN INT)

Figure 10-25. Interrupt Register (DCAN INT) [offset = 0x10]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								Int1ID[7:0]							
R-0								R-0							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Int0ID[15:0]															
R-0															

R = Read, -*n* = Value after reset

Table 10-8. Interrupt Register Field Descriptions

Bit	Name	Value	Description
31-24	Reserved		These bits are always read as 0. Writes have no effect.
23-16	Int1ID[23:16]		Interrupt 1 Identifier (indicates the message object with the highest pending interrupt)
		0x00	No interrupt is pending
		0x01-0x80	Number of message object which caused the interrupt.
		0x81-0xFF	Unused
			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.

Table 10-8. Interrupt Register Field Descriptions

Bit	Name	Value	Description
15-0	Int0ID[15:0]		Interrupt Identifier (the number here indicates the source of the interrupt)
		0x0000	No interrupt is pending
		0x0001-0x0080	Number of message object which caused the interrupt.
		0x0081-0x7FFF	Unused
		0x8000	Error and Status Register value is not 0x07.
		0x8001-0xFFFF	Unused
			If several interrupts are pending, the CAN Interrupt Register will point to the pending interrupt with the highest priority. The 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.

10.17.6 Test Register (DCAN TEST)

Figure 10-26. Test Register (DCAN TEST) [offset = 0x14]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						RDA	EXL	Rx	Tx[1:0]		LBack	Silent	Reserved		
R-0						R/WP-0	R/WP-0	R-U	R/WP-0		R/WP-0	R/WP-0	R-0		

R = Read, WP = Write Protected by Test bit, -*n* = Value after reset, -U = Undefined

.

Table 10-9. Test Register Field Descriptions

Bit	Name	Value	Description
31–10	Reserved		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[1:0]		Control of CAN_TX pin
		00	Normal operation, CAN_TX is controlled by the CAN Core.
		01	Sample Point can be monitored at CAN_TX pin.
		10	CAN_TX pin drives a dominant value.
		11	CAN_TX pin drives a recessive value.
4	LBack		Loop Back Mode
		0	Disabled
		1	Enabled

Table 10-9. Test Register Field Descriptions

Bit	Name	Value	Description
3	Silent		Silent Mode
		0	Disabled
		1	Enabled
2-0	Reserved		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 one. 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 Test bit is cleared.

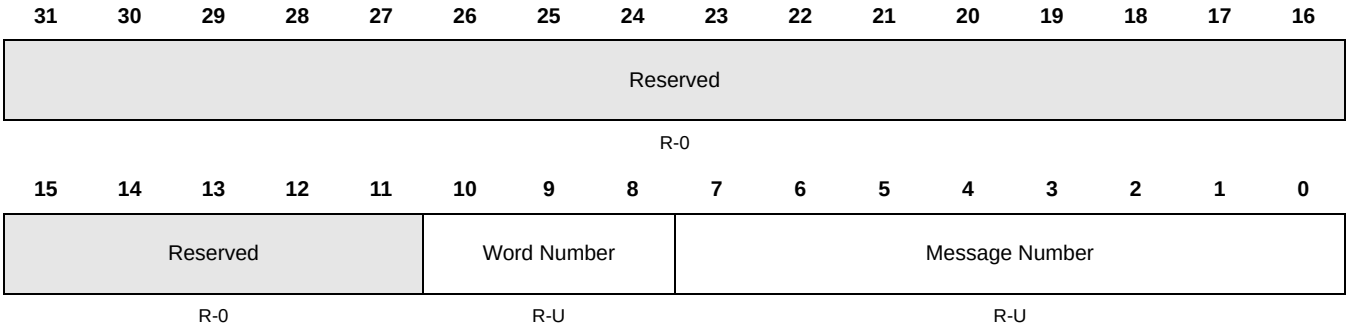
Note:
The Test Register is only writable if Test bit in CAN Control Register is set.

Note:
Setting Tx[1:0] other than ‘00’ will disturb message transfer.

Note:
When the internal loop back mode is active (bit LBack is set), bit EXL will be ignored.

10.17.7 Parity Error Code Register (DCAN PERR)

Figure 10-27. Parity Error Code Register (DCAN PERR) [offset = 0x1C]



R = Read, *-n* = Value after reset, -U = Undefined

.

Table 10-10. Parity Error Code Register Field Descriptions

Bit	Name	Value	Description
31–11	Reserved		These bits are always read as 0. Writes have no effect.
10-8	Word Number	0x01-0x05	Word number where parity error has been detected RDA word number (1 to 5) of the message object (according to the Message RAM representation in RDA mode, see section 10.16.4).
7–0	Message Number	0x01-0x80	Message object number where parity error has been detected

If a parity error is detected, the PER flag will be set in the Error and Status Register. This bit is not reset by the parity check mechanism; it must be reset by reading the Error and Status Register.

In addition to the PER flag, the Parity Error Code Register will indicate the memory area where the parity error has been detected (message number and word number).

If more than one word with a parity error was detected, the highest word number with a parity error will be displayed.

After a parity error has been detected, the register will hold the last error code until power is removed.

10.17.8 DCAN Core Release Register (DCAN REL)

Figure 10-28. DCAN Core Release Register (DCAN REL) [offset = 0x20]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REL				STEP				SUBSTEP				YEAR			
R-0xA				R-0x3				R-0x1				R-0x7			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MON								DAY							
R-0x05								R-0x04							

R = Read, *-n* = Value after reset; D = device dependent

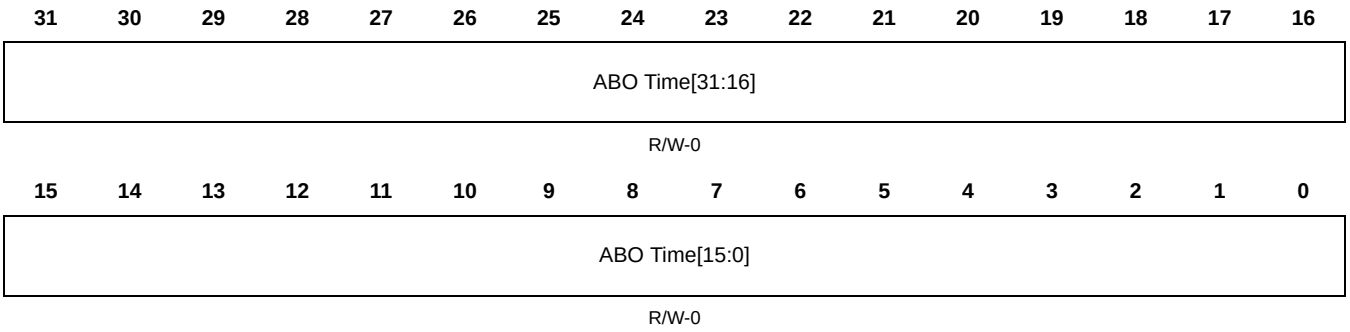
.

Table 10-11. DCAN Core Release Register Field Descriptions

Bit	Name	Value	Description
31–28	REL	0xA	Core Release One digit, BCD-coded.
27-24	STEP	0x3	Step of Core Release One digit, BCD-coded.
23-20	SUBSTEP	0x1	Substep of Core Release One digit, BCD-coded.
19-16	YEAR	0x7	Design Time Stamp, Year One digit, BCD-coded. This field is set by constant parameter on DCAN synthesis.
15-8	MON	0x05	Design Time Stamp, Month Two digits, BCD-coded. This field is set by constant parameter on DCAN synthesis.
7-0	DAY	0x04	Design Time Stamp, Day Two digits, BCD-coded. This field is set by constant parameter on DCAN synthesis.

10.17.9 Auto-Bus-On Time Register (DCAN ABOTR)

Figure 10-29. Auto-Bus-On Time Register (DCAN ABOTR) [offset = 0x80]



R = Read, W = Write, -*n* = Value after reset

.

Table 10-12. Auto-Bus-On Time Register Field Descriptions

Bit	Name	Value	Description
31–0	ABO Time		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 bit ABO in CAN Control Register. The Auto-Bus-On timer is realized by a 32-bit counter which starts to count down to zero when the module goes Bus-Off. The counter will be reloaded with the preload value of the ABO Time register after this phase.

Note:
On write access to the CAN Control register while Auto-Bus-On timer is running, the Auto-Bus-On procedure will be aborted.

Note:
During Debug/Suspend mode, running Auto-Bus-On timer will be paused.

10.17.10Transmission 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 10-30. Transmission Request X Register (DCAN TXRQ X) [offset = 0x84]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TxRqstReg8		TxRqstReg7		TxRqstReg6		TxRqstReg5		TxRqstReg4		TxRqstReg3		TxRqstReg2		TxRqstReg1	
R-0		R-0		R-0		R-0		R-0		R-0		R-0		R-0	

R = Read, *-n* = Value after reset

Example:

Bit 0 of the Transmission Request X Register represents byte 0 of the Transmission Request 1 Register. If one or more bits in this byte are set, bit 0 of the Transmission Request X Register will be set.

10.17.11Transmission 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 10-31. Transmission Request Registers [offset = 0x88 to 0x94]

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
0x88 CAN TXRQ12	TxRqst[32:17]															
	R-0															
	TxRqst[16:1]															
	R-0															
0x8C CAN TXRQ34	TxRqst[64:49]															
	R-0															
	TxRqst[48:33]															
	R-0															
0x90 CAN TXRQ56	TxRqst[96:81]															
	R-0															
	TxRqst[80:65]															
	R-0															
0x94 CAN TXRQ78	TxRqst[128:113]															
	R-0															
	TxRqst[112:97]															
	R-0															
	R = Read, *-n* = Value after reset															

Table 10-13. Transmission Request Registers Field Descriptions

Bit	Name	Value	Description
31-0	TxRqs[128:1]		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.

10.17.12 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 10-32. New Data X Register (DCAN NWDAT X) [offset = 0x98]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NewDatReg8		NewDatReg7		NewDatReg6		NewDatReg5		NewDatReg4		NewDatReg3		NewDatReg2		NewDatReg1	
R-0		R-0		R-0		R-0		R-0		R-0		R-0		R-0	

R = Read, *-n* = Value after reset

Example:

Bit 0 of the New Data X Register represents byte 0 of the New Data 1 Register. If one or more bits in this byte are set, bit 0 of the New Data X Register will be set.

10.17.13New 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 10-33. New Data Registers [offset = 0x9C to 0xA8]

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
0x9C CAN NWDAT12	NewDat[32:17]															
	R-0															
	NewDat[16:1]															
	R-0															
0xA0 CAN NWDAT34	NewDat[64:49]															
	R-0															
	NewDat[48:33]															
	R-0															
0xA4 CAN NWDAT56	NewDat[96:81]															
	R-0															
	NewDat[80:65]															
	R-0															
0xA8 CAN NWDAT78	NewDat[128:113]															
	R-0															
	NewDat[112:97]															
	R-0															
	R = Read, *-n* = Value after reset															

Table 10-14. New Data Registers Field Descriptions

Bit	Name	Value	Description
31-0	NewDat[128:1]		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.

10.17.14 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 10-34. Interrupt Pending X Register (DCAN INTPND X) [offset = 0xAC]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IntPndReg8		IntPndReg7		IntPndReg6		IntPndReg5		IntPndReg4		IntPndReg3		IntPndReg2		IntPndReg1	
R-0		R-0		R-0		R-0		R-0		R-0		R-0		R-0	

R = Read, *-n* = Value after reset

Example:

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.

10.17.15Interrupt 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 10-35. Interrupt Pending Registers [offset = 0xB0 to 0xBC]

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
0xB0 CAN INTPND12	IntPnd[32:17]															
	R-0															
	IntPnd[16:1]															
	R-0															
0xB4 CAN INTPND34	IntPnd[64:49]															
	R-0															
	IntPnd[48:33]															
	R-0															
0xB8 CAN INTPND56	IntPnd[96:81]															
	R-0															
	IntPnd[80:65]															
	R-0															
0xBC CAN INTPND78	IntPnd[128:113]															
	R-0															
	IntPnd[112:97]															
	R-0															
	R = Read, *-n* = Value after reset															

Table 10-15. Interrupt Pending Registers Field Descriptions

Bit	Name	Value	Description
31-0	IntPnd[128:1]		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.

10.17.16Message 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 10-36. Message Valid X Register (DCAN MSGVAL X) [offset = 0xC0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MsgValReg8		MsgValReg7		MsgValReg6		MsgValReg5		MsgValReg4		MsgValReg3		MsgValReg2		MsgValReg1	
R-0		R-0		R-0		R-0		R-0		R-0		R-0		R-0	

R = Read, *-n* = Value after reset

Example:

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.

10.17.17Message 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 10-37. Message Valid Registers [offset = 0xC4 to 0xD0]

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
0xC4 CAN MSGVAL12	MsgVal[32:17]															
	R-0															
	MsgVal[16:1]															
	R-0															
0xC8 CAN MSGVAL34	MsgVal[64:49]															
	R-0															
	MsgVal[48:33]															
	R-0															
0xCC CAN MSGVAL56	MsgVal[96:81]															
	R-0															
	MsgVal[80:65]															
	R-0															
0xD0 CAN MSGVAL78	MsgVal[128:113]															
	R-0															
	MsgVal[112:97]															
	R-0															
	R = Read, *-n* = Value after reset															

Table 10-16. Message Valid Registers Field Descriptions

Bit	Name	Value	Description
31-0	MsgVal[128:1]		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.

10.17.18Interrupt Multiplexer Registers (DCAN INTMUX12 to DCAN INTMUX78)

The IntMux flag determine 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 10-38. Interrupt Multiplexer Registers [offset = 0xD8 to 0xE4]

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
0xD8 CAN INTMUX12	IntMux[31:16]															
	R/W-0															
	IntMux[15:0]															
	R/W-0															
0xDC CAN INTMUX34	IntMux[63:48]															
	R/W-0															
	IntMux[47:32]															
	R/W-0															
0xE0 CAN INTMUX56	IntMux[95:80]															
	R/W-0															
	IntMux[79:64]															
	R/W-0															
0xE4 CAN INTMUX78	IntMux[127:112]															
	R/W-0															
	IntMux[111:96]															
	R/W-0															
	R = Read, W = Write, *-n* = Value after reset															

Table 10-17. Interrupt Multiplexer Registers Field Descriptions

Bit	Name	Value	Description
31-0	IntMux[127:0]		Multiplexes IntPnd value to either DCAN0INT or DCAN1INT interrupt lines (for all message objects).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 one.
		1	DCAN1INT line is active if corresponding IntPnd flag is one.

10.17.19IF1/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 Busy bit is one, IF1/IF2 Register sets are write protected.

Note:
For debug support, the auto clear functionality of the IF1/IF2 Command Registers (clear of DMAactive flag by r/w) is disabled during Debug/Suspend mode.

Note:
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 10-39. IF1 Command Registers (DCAN IF1CMD) [offset = 0x100]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								WR/RD	Mask	Arb	Control	Clr IntPnd	TxRqst/ NewDat	Data A	Data B
R-0								R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Busy	DMA active	Reserved						Message Number							
R-0	R/WP/C-0	R-0						R/WP-0x1							

R = Read, WP = Protected Write (protected by Busy bit), C = Clear by IF1 access, -*n* = Value after reset

Figure 10-40. IF2 Command Registers (CAN IF2CMD) [offset = 0x120]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								WR/RD	Mask	Arb	Control	Clr IntPnd	TxRqst/ NewDat	Data A	Data B
R-0								R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Busy	DMA active	Reserved						Message Number							
R-0	R/WP/C-0	R-0						R/WP-0x1							

R = Read, WP = Protected Write (protected by Busy bit), C = Clear by IF2 access, -*n* = Value after reset

Table 10-18. IF1/IF2 Command Register Field Descriptions

Bit	Name	Value	Description
31–24	Reserved		These bits are always read as 0. Writes have no effect.

Bit	Name	Value	Description
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 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 only be controlled by the Control flag (Bit [20]).

Bit	Name	Value	Description
18	TxRqst/NewDat		Access Transmission Request Bit
		0	Direction = Read: NewDat bit will not be changed. Direction = Write: TxRqst/NewDat bit will be handled according to the Control bit.
		1	Direction = Read: Clears NewDat bit in the message object. Direction = Write: Sets TxRqst/NewDat in message object.
			Note: If a CAN transmission is requested by setting TxRqst/NewDat in this register, the TxRqst/NewDat bits in the message object will be set to one independent of the values in IF1/IF2 Message Control Register.
			Note: A read access to a message object can be combined with the reset of the control bits IntPnd and NewDat. The values of these bits transferred to the IF1/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.
16	Data B		Access Data Bytes 4-7
		0	Data Bytes 4-7 will not be changed.
		1	Direction = Read: The Data Bytes 4-7 will be transferred from the message object addressed by Message Number (Bits [7:0]) to the corresponding IF1/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 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 one 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 been finished.

Bit	Name	Value	Description
14	DMAactive		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 DMAactive is one. Note: Due to the auto reset feature of the DMAactive bit, this bit has to be set for each subsequent DMA cycle separately.
13-8	Reserved		These bits are always read as 0. Writes have no effect.
7–0	Message Number		Number of message object in Message RAM which is used for data transfer
		0x00	Invalid message number
		0x01-0x80	Valid message numbers
		0x81-0xFF	Invalid message numbers

10.17.20IF1/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 10.16.1.

Note:
While Busy bit of IF1/IF2 Command Register is one, IF1/IF2 Register Set is write protected.

Figure 10-41. IF1 Mask Register (DCAN IF1MSK) [offset = 0x104]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MXtd	MDir	Reserved	Msk[28:16]												
R/WP-1	R/WP-1	R-1	R/WP-0x1FFF												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Msk[15:0]															
R/WP-0xFFFF															

R = Read, WP = Protected Write (protected by Busy bit), -*n* = Value after reset

Figure 10-42. IF2 Mask Register (DCAN IF2MSK) [offset = 0x124]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MXtd	MDir	Reserved	Msk[28:16]												
R/WP-1	R/WP-1	R-1	R/WP-0x1FFF												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Msk[15:0]															
R/WP-0xFFFF															

R = Read, WP = Protected Write (protected by Busy bit), -*n* = Value after reset

.

Table 10-19. IF1/IF2 Mask Registers Field Descriptions

Bit	Name	Value	Description
31	MXtd		Mask Extended Identifier
		0	The extended identifier bit (IDE) has no effect on the acceptance filtering.
		1	The extended identifier bit (IDE) is used for acceptance filtering.
			When 11-bit (“standard”) identifiers are used for a message object, the identifiers of received Data Frames are written into bits ID[28:18]. For acceptance filtering, only these bits together with mask bits Msk[28:18] are considered.

Table 10-19. IF1/IF2 Mask Registers Field Descriptions

Bit	Name	Value	Description
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		These bits are always read as 1. Writes have no effect.
28-0	Msk[28:0]		Identifier Mask
		0	The corresponding bit in the identifier of the message object is not used for acceptance filtering (don't care).
		1	The corresponding bit in the identifier of the message object is used for acceptance filtering.

10.17.21IF1/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 10.16.1

Note:
While Busy bit of IF1/IF2 Command Register is one, IF1/IF2 Register Set is write protected.

Figure 10-43. IF1 Arbitration Register (DCAN IF1ARB) [offset = 0x108]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MsgVal	Xtd	Dir	ID[28:16]												
R/WP-0	R/WP-0	R/WP-0	R/WP-0												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
R/WP-0															

R = Read, WP = Protected Write (protected by Busy bit), *-n* = Value after reset

Figure 10-44. IF2 Arbitration Register (DCAN IF2ARB) [offset = 0x128]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MsgVal	Xtd	Dir	ID[28:16]												
R/WP-0	R/WP-0	R/WP-0	R/WP-0												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
R/WP-0															

R = Read, WP = Protected Write (protected by Busy bit), *-n* = Value after reset

.

Table 10-20. IF1/IF2 Arbitration Registers Field Descriptions

Bit	Name	Value	Description
31	MsgVal		Message Valid
		0	The message object is ignored by the Message Handler.
		1	The message object is to be used by the Message Handler. The CPU should reset the MsgVal bit of all unused Messages Objects during the initialization before it resets bit Init in the CAN Control Register. This bit must also be reset before the identifier ID[28:0], the control bits Xtd, Dir, DLC[3:0], RxIE, TxIE, RmtEn, EoB, UMask, the mask bits Msk28:0, MXtd, and MDir are modified, or if the messages object is no longer required.

Table 10-20. IF1/IF2 Arbitration Registers Field Descriptions

Bit	Name	Value	Description
30	Xtd		Extended Identifier
		0	The 11-bit (“standard”) Identifier is used for this message object.
		1	The 29-bit (“extended”) Identifier is used for this message object.
29	Dir		Message Direction
		0	Direction = receive: On TxRqst, a Remote Frame with the identifier of this message object is transmitted. On reception of a Data Frame with matching identifier, this message is stored in this message object.
		1	Direction = transmit: On TxRqst, the respective message object is transmitted as a Data Frame. On reception of a Remote Frame with matching identifier, the TxRqst bit of this message object is set (if RmtEn = 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 = one, standard frames in message objects with Xtd = zero.

If a received message (Data Frame or Remote Frame) matches more than one valid message objects, it is stored into the one with the lowest message number.

10.17.22IF1/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 10.16.1.

Note:
While Busy bit of IF1/IF2 Command Register is one, IF1/IF2 Register Set is write protected.

Figure 10-45. IF1 Message Control Register (DCAN IF1MCTL) [offset = 0x10C]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
New Dat	Msg Lst	Int Pnd	UMask	TxIE	RxIE	Rmt En	Tx Rqst	EoB	Reserved			DLC[3:0]			
R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R-0			R/WP-0			

R = Read, WP = Protected Write (protected by Busy bit), *-n* = Value after reset

Figure 10-46. IF2 Message Control Register (DCAN IF2MCTL) [offset = 0x12C]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
New Dat	Msg Lst	Int Pnd	UMask	TxIE	RxIE	Rmt En	Tx Rqst	EoB	Reserved			DLC[3:0]			
R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R-0			R/WP-0			

R = Read, WP = Protected Write (protected by Busy bit), *-n* = Value after reset

.

Table 10-21. IF1 / IF2 Message Control Registers Field Descriptions

Bit	Name	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 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.

Table 10-21. IF1 / IF2 Message Control Registers Field Descriptions

Bit	Name	Value	Description
14	MsgLst		Message Lost (only valid for message objects with direction = receive)
		0	No message lost since the last time when this bit was reset by the CPU.
		1	The Message Handler stored a new message into this object when NewDat was still set, so the previous message has been overwritten.
13	IntPnd		Interrupt Pending
		0	This message object is not the source of an interrupt.
		1	This message object is the source of an interrupt. The Interrupt Identifier in the Interrupt Register will point to this message object if there is no other interrupt source with higher priority.
12	UMask		Use Acceptance Mask
		0	Mask ignored
		1	Use Mask (Msk[28:0], MXtd, and MDir) for acceptance filtering
			If the UMask bit is set to one, the message object's mask bits have to be programmed during initialization of the message object before MsgVal is set to one.
11	TxIE		Transmit Interrupt Enable
		0	IntPnd will not be triggered after the successful transmission of a frame.
		1	IntPnd will be triggered after the successful transmission of a frame.
10	RxIE		Receive Interrupt Enable
		0	IntPnd will not be triggered after the successful reception of a frame.
		1	IntPnd will be triggered after the successful reception of a frame.
9	RmtEn		Remote Enable
		0	At the reception of a Remote Frame, TxRqst is not changed.
		1	At the reception of a Remote Frame, TxRqst is set.
			Note: See section 10.13.8 for details on the setup of RmtEn and UMask for remote frames.
8	TxRqst		Transmit Request
		0	This message object is not waiting for a transmission.
		1	The transmission of this message object is requested and is not yet done.

Table 10-21. IF1 / IF2 Message Control Registers Field Descriptions

Bit	Name	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 one.
6-5	Reserved		These bits are always read as 0. Writes have no effect.
3-0	DLC[3:0]		Data Length Code
		0-8	Data Frame has 0-8 data bits.
		9-15	Data Frame has 8 data bytes.
			Note: The Data Length Code of a message object must be defined the same as in all the corresponding objects with the same identifier at other nodes. When the Message Handler stores a data frame, it will write the DLC to the value given by the received message.

10.17.23IF1/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 10-47. IF1 Data A Register (DCAN IF1DATA) [offset = 0x110]

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

R = Read, WP = Protected Write (protected by Busy bit), -*n* = Value after reset

Figure 10-48. IF1 Data B Register (DCAN IF1DATB) [offset = 0x114]

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

R = Read, WP = Protected Write (protected by Busy bit), -*n* = Value after reset

Figure 10-49. IF2 Data A Register (DCAN IF2DATA) [offset = 0x130]

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

R = Read, WP = Protected Write (protected by Busy bit), -*n* = Value after reset

Figure 10-50. IF2 Data B Register (DCAN IF2DATB) [offset = 0x134]

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

R = Read, WP = Protected Write (protected by Busy bit), *-n* = Value after reset

10.17.24IF3 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 (see also section 10.16.1).

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]).

If an interrupt line is available for IF3, an interrupt will be generated by IF3Upd flag. Please refer to the device datasheet 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 10-51. IF3 Observation Register (DCAN IF3OBS) [offset = 0x140]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IF3 Upd	Reserved		IF3 SDB	IF3 SDA	IF3 SC	IF3 SA	IF3 SM	Reserved			Data B	DataA	Ctrl	Arb	Mask
R-0	R-0		R-0	R-0	R-0	R-0	R-0	R-0			R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

R = Read, W = Write, WP = Protected Write, S = Set, C = Clear by Read, U = Undefined, -*n* = Value after reset

Table 10-22. IF3 Observation register Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		These bits are always read as 0. Writes have no effect.
15	IF3 Upd		IF3 Updata Data
		0	No new data has been loaded since last IF3 read.
		1	New data has been loaded since last IF3 read.
14–13	Reserved		These bits are always read as 0. Writes have no effect.

Table 10-22. IF3 Observation register Field Descriptions

Bit	Name	Value	Description
12	IF3 SDB		IF3 Status of Data B read access
		0	All Data B bytes are already read out, or are not marked to be read.
		1	Data B section has still data to be read out.
11	IF3 SDA		IF3 Status of Data A read access
		0	All Data A bytes are already read out, or are not marked to be read.
		1	Data A section has still data to be read out.
10	IF3 SC		IF3 Status of Control bits read access
		0	All Control section bytes are already read out, or are not marked to be read.
		1	Control section has still data to be read out.
9	IF3 SA		IF3 Status of Arbitration data read access
		0	All Arbitration data bytes are already read out, or are not marked to be read.
		1	Arbitration section has still data to be read out.
8	IF3 SM		IF3 Status of Mask data read access
		0	All Mask data bytes are already read out, or are not marked to be read.
		1	Mask section has still data to be read out.
7–5	Reserved		These bits are always read as 0. Writes have no effect.
4	DataB		Data B read observation
		0	Data B section has not to be read.
		1	Data B section has to be read to enable next IF3 update.
3	DataA		Data A read observation
		0	Data A section has not to be read.
		1	Data A section has to be read to enable next IF3 update.
2	Ctrl		Ctrl read observation
		0	Ctrl section has not to be read.
		1	Ctrl section has to be read to enable next IF3 update.
1	Arb		Arbitration data read observation
		0	Arbitration data has not to be read.
		1	Arbitration data has to be read to enable next IF3 update.

Table 10-22. IF3 Observation register Field Descriptions

Bit	Name	Value	Description
0	Mask		Mask data read observation
		0	Mask data has not to be read.
		1	Mask data has to be read to enable next IF3 update.

10.17.25IF3 Mask Register (DCAN IF3MSK)

Figure 10-52. IF3 Mask Register (DCAN IF3MSK) [offset = 0x144]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MXtd	MDir	Reserved	Msk[28:16]												
R-1	R-1	R-1	R-0x1FFF												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Msk[15:0]															
R-0xFFFF															

R = Read, *-n* = Value after reset

Table 10-23. IF3 Mask Register Field Descriptions

Bit	Name	Value	Description
31	MXtd		Mask Extended Identifier
		0	The extended identifier bit (IDE) has no effect on the acceptance filtering.
		1	The extended identifier bit (IDE) is used for acceptance filtering.
			Note: When 11-bit (“standard”) Identifiers are used for a message object, the identifiers of received Data Frames are written into bits ID[28:18]. For acceptance filtering, only these bits together with mask bits Msk[28:18] are considered.
30	MDir		Mask Massage 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		These bits are always read as 0. Writes have no effect.
28-0	Msk[28:0]		Identifier Mask
		0	The corresponding bit in the identifier of the message object is not used for acceptance filtering (don’t care).
		1	The corresponding bit in the identifier of the message object is used for acceptance filtering.

10.17.26IF3 Arbitration Register (DCAN IF3ARB)

Figure 10-53. IF3 Arbitration Register (DCAN IF3ARB) [offset = 0x148]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MsgVal	Xtd	Dir	ID[28:16]												
R-0	R-0	R-0	R-0												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
R-0															

R = Read, *-n* = Value after reset

Table 10-24. IF3 Arbitration Register Field Descriptions

Bit	Name	Value	Description
31	MsgVal		Message Valid
		0	The message object is ignored by the Message Handler.
		1	The message object is to be used by the Message Handler.
			The CPU should reset the MsgVal bit of all unused Messages Objects during the initialization before it resets bit Init in the CAN Control Register. This bit must also be reset before the identifier ID[28:0], the control bits Xtd, Dir or DLC[3:0] are modified, or if the messages object is no longer required.
30	Xtd		Extended Identifier
		0	The 11-bit (“standard”) Identifier is used for this message object.
		1	The 29-bit (“extended”) Identifier is used for this message object.
29	Dir		Message Direction
		0	Direction = receive: On TxRqst, a Remote Frame with the identifier of this message object is transmitted. On reception of a Data Frame with matching identifier, that message is stored in this message object.
		1	Direction = transmit: On TxRqst, the respective message object is transmitted as a Data Frame. On reception of a Remote Frame with matching identifier, the TxRqst bit of this message object is set (if RmtEn = one).
28-0	ID[28:0]		Message Identifier
		ID[28:0]	29-bit Identifier (“Extended Frame”)
		ID[28:18]	11-bit Identifier (“Standard Frame”)

10.17.27IF3 Message Control Register (DCAN IF3MCTL)

Figure 10-54. IF3 Message Control Register (DCAN IF3MCTL) [offset = 0x14C]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
New Dat	Msg Lst	Int Pnd	UMask	TxIE	RxIE	Rmt En	Tx Rqst	EoB	Reserved			DLC[3:0]			
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0			R-0			

R = Read, *-n* = Value after reset

Table 10-25. IF3 Message Control Register Field Descriptions

Bit	Name	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 when this flag was cleared by the CPU.
		1	The Message Handler or the CPU has written new data into the data portion of this message object.
14	MsgLst		Message Lost (only valid for message objects with direction = receive)
		0	No message lost since the last time when this bit was reset by the CPU.
		1	The Message Handler stored a new message into this object when NewDat was still set, so the previous message has been overwritten.
13	IntPnd		Interrupt Pending
		0	This message object is not the source of an interrupt.
		1	This message object is the source of an interrupt. The Interrupt Identifier in the Interrupt Register will point to this message object if there is no other interrupt source with higher priority.
12	UMask		Use Acceptance Mask
		0	Mask ignored
		1	Use Mask (Msk[28:0], MXtd, and MDir) for acceptance filtering
			If the UMask bit is set to one, the message object's mask bits have to be programmed during initialization of the message object before MsgVal is set to one.

Table 10-25. IF3 Message Control Register Field Descriptions

Bit	Name	Value	Description
11	TxIE		Transmit Interrupt Enable
		0	IntPnd will not be triggered after the successful transmission of a frame.
		1	IntPnd will be triggered after the successful transmission of a frame.
10	RxIE		Receive Interrupt Enable
		0	IntPnd will not be triggered after the successful reception of a frame.
		1	IntPnd will be triggered after the successful reception of a frame.
9	RmtEn		Remote Enable
		0	At the reception of a Remote Frame, TxRqst is not changed.
		1	At the reception of a Remote Frame, TxRqst is set.
			Note: See section 10.13.8 for details on the setup of RmtEn and UMask for remote frames.
8	TxRqst		Transmit Request
		0	This message object is not waiting for a transmission.
		1	The transmission of this message object is requested and is not yet done.
7	EoB		End of Block
		0	The message object is part of a FIFO Buffer block and is not the last message object of the FIFO Buffer block.
		1	The message object is a single message object or the last message object in a FIFO Buffer Block.
			Note: This bit is used to concatenate multiple message objects to build a FIFO Buffer. For single message objects (not belonging to a FIFO Buffer), this bit must always be set to one.
6-4	Reserved		These bits are always read as 0. Writes have no effect.
3-0	DLC[3:0]		Data Length Code
		0-8	Data Frame has 0-8 data bits.
		9-15	Data Frame has 8 data bytes.
			Note: The Data Length Code of a message object must be defined the same as in all the corresponding objects with the same identifier at other nodes. When the Message Handler stores a data frame, it will write the DLC to the value given by the received message.

10.17.28IF3 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 10-55. IF3 Data A Register (DCAN IF3DATA) [offset = 0x150]

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

R = Read, *-n* = Value after reset

Figure 10-56. IF3 Data A Register (DCAN IF3DATB) [offset = 0x154]

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

R = Read, *-n* = Value after reset

10.17.29IF3 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 (e.g due to reception of a CAN frame) will trigger an automatic copy of the whole message object to IF3 register set.

Note:
IF3 Update enable should not be set for transmit objects.

Figure 10-57. IF3 Update Enable Registers [offset = 0x160 to 0x16C]

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
0x160 CAN IF3UPD12	IF3UpdEn[32:17]															
	R/W-0															
	IF3UpdEn[16:1]															
	R/W-0															
0x164 CAN IF3UPD34	IF3UpdEn[64:49]															
	R/W-0															
	IF3UpdEn[48:33]															
	R/W-0															
0x168 CAN IF3UPD56	IF3UpdEn[96:81]															
	R/W-0															
	IF3UpdEn[80:65]															
	R/W-0															
0x16C CAN IF3UPD78	IF3UpdEn[128:113]															
	R/W-0															
	IF3UpdEn[112:97]															
	R/W-0															
	R = Read, W = Write, *-n* = Value after reset															

Table 10-26. IF3 Update Control Register Field Descriptions

Bit	Name	Value	Description
31-0	IF3UpdEn[128:1]		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.

10.17.30CAN 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.

Note:
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 10-58. CAN TX IO Control Register (DCAN TIOC) [offset = 0x1E0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												Res	PU	PD	OD
R-0												R-0	R/W-D	R/W-D	R/WP-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												Func	Dir	Out	In
R-0												R/WP-0	R/WP-0	R/WP-0	R-U

R = Read, W = Write, WP = Protected Write (protected by Init bit), -*n* = Value after reset; D = device dependent

.

Table 10-27. CAN TX IO Control Register Field Descriptions

Bit	Name	Value	Description
31–20	Reserved	0	These bits are always read as 0. Writes have no effect.
19	Reserved	0	Do NOT write any value other than ‘0’ to this bit.
18	PU		Selection of pull direction. Pull function has to be enabled by resetting bit PD.
		0	Pull down
		1	Pull up
17	PD		Pull functionality disable
		0	Pull functionality is enabled
		1	Pull functionality is disabled
16	OD		Open drain mode
		0	Push Pull output driver
		1	Open Drain output driver
			Forced to ‘0’ if Init bit of CAN Control register is reset.
15-4	Reserved		These bits are always read as 0. Writes have no effect.

Table 10-27. CAN TX IO Control Register Field Descriptions

Bit	Name	Value	Description
3	Func		Functionality of CAN_TX pin
		0	Used as general purpose IO pin.
		1	Used as functional pin (transmission of serial CAN data).
			Forced to '1' if Init bit of CAN Control register is reset.
2	Dir		Direction of pin if configured as general purpose IO pin
		0	Used as input pin.
		1	Output driver is enabled.
			Forced to '1' if Init bit of CAN Control register is reset.
1	Out		Value to drive to CAN_TX output buffer if configured as general purpose IO pin
		0	Output buffer is driven low.
		1	Output buffer is driven high.
			Forced to Tx output of the CAN Core, if Init bit of CAN Control register is reset.
0	In		Input value of CAN_TX pin
		0	Pin level is low.
		1	Pin level is high.
			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 (e.g. while reset of the DCAN module).

10.17.31CAN 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.

Note:
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 10-59. CAN RX IO Control Register (DCAN RIOC) [offset = 0x1E4]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												Res	PU	PD	OD
R-0												R-0	R/W-D	R/W-D	R/WP-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												Func	Dir	Out	In
R-0												R/WP-0	R/WP-0	R/W-0	R-U

R = Read, W = Write, WP = Protected Write (protected by Init bit), -*n* = Value after reset; D = device dependent

Table 10-28. CAN RX IO Control Register Field Descriptions

Bit	Name	Value	Description
31–20	Reserved	0	These bits are always read as 0. Writes have no effect.
19	Reserved	0	Do NOT write any value other than ‘0’ to this bit.
18	PU		Selection of pull direction. Pull function has to be enabled by resetting bit PD.
		0	Pull down
		1	Pull up
17	PD		Pull functionality disable
		0	Pull functionality is enabled
		1	Pull functionality is disabled
16	OD		Open drain mode
		0	Push Pull output driver
		1	Open Drain output driver
			Forced to ‘0’ if Init bit of CAN Control register is reset.
15-4	Reserved		These bits are always read as 0. Writes have no effect.

Table 10-28. CAN RX IO Control Register Field Descriptions

Bit	Name	Value	Description
3	Func		Functionality of CAN_RX pin
		0	Used as general purpose IO pin.
		1	Used as functional pin (reception of serial CAN data).
			Forced to '1' if Init bit of CAN Control register is reset.
2	Dir		Direction of pin if configured as general purpose IO pin
		0	Used as input pin.
		1	Output driver is enabled.
			Forced to '0' if Init bit of CAN Control register is reset.
1	Out		Value to drive to CAN_RX output buffer if configured as general purpose IO pin
		0	Output buffer is driven low.
		1	Output buffer is driven high.
0	In		Input value of CAN_RX pin
		0	Pin level is low.
		1	Pin level is high.
			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 (e.g. while reset of the DCAN module).

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.

Topic **Page**

11.1 Overview 446
11.2 Operating Modes 448
11.3 Test Features 467
11.4 General-Purpose I/O 469
11.5 Low-Power Mode 470
11.6 Interrupts 471
11.7 DMA Interface 474
11.8 Module Configuration 475
11.9 Control Registers 477
11.10 Multi-Buffer RAM 569
11.11 Parity Memory 579
11.12 MibSPI Pin Timing Parameters 582

11.1 Overview

The SPI/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
- 8-bit baud clock generator, supports max up to 11Mhz baud rate
- Serial clock (SPICLK) pin
- Up to 4 SPISOMI/SPISIMO pins for data transfer, with programmable pin direction
- SPI enable ($\overline{\text{SPIENA}}$) pin
- Up to 7 slave chip select ($\overline{\text{SPISCS}}$[7:0]) 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 Data Pin
SOMI - Slave Out Master In Data Pin
CS - SPI Chip Select Pin
SPICLK - SPI Clock Pin
ENA - SPI Enable Pin
Refer to the device specific data sheet for more details on the implemetation of the SPI, MibSPI and MibSPIP modules.

11.1.1 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

11.1.2 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, etc.) 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.

11.1.2.1 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).

11.1.2.2 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.

11.1.3 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.

11.2 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 ($\overline{\text{SPISCS}}$[7:0]) pins, are used when communicating with multiple slave devices. When the a write occurs to SPIDAT1 in master mode, the $\overline{\text{SPISCS}}$ 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.

11.2.1 Pin Configurations

The SPI supports data connections as shown in Table 11-1.

Table 11-1. Pin Configurations

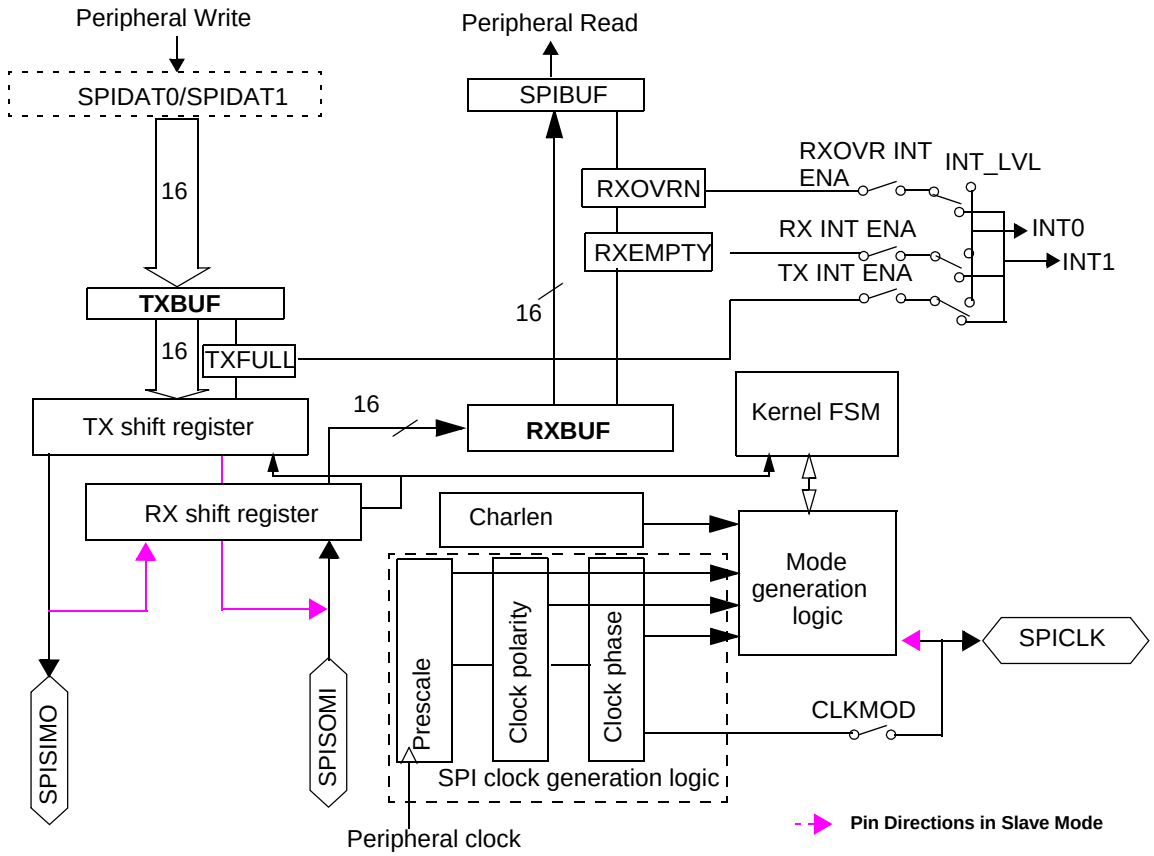
Pin	Master Mode		Slave Mode	
SPICLK	Drives the clock to external devices		Receives the clock from the external master	
SPISOMI	Receives data from the external slave		Sends data to the external master	
SPISIMO	Transmits data to the external slave		Receives data from the external master	
SPIENA	**SPIENA disabled:** GIO	**SPIENA enabled:** Receives ENA signal from the external slave	**SPIENA disabled:** GIO	**SPIENA enabled:** Receives ENA signal from the external master
SPICS[7:0]	**SPICS disabled:** GIO	**SPICS enabled:** Selects one or more slave devices	**SPICS disabled:** GIO	**SPICS enabled:** Receives the CS signal from the external master

Note:
1) When the SPICS[7:0] signals are disabled, the chip-select field in the transmit data is not used.
2) When the 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.

11.2.2 Data Handling

Figure 11-1 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.

Figure 11-1. 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.

11.2.2.1 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.

11.2.2.2 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.

11.2.2.1 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 11-2).

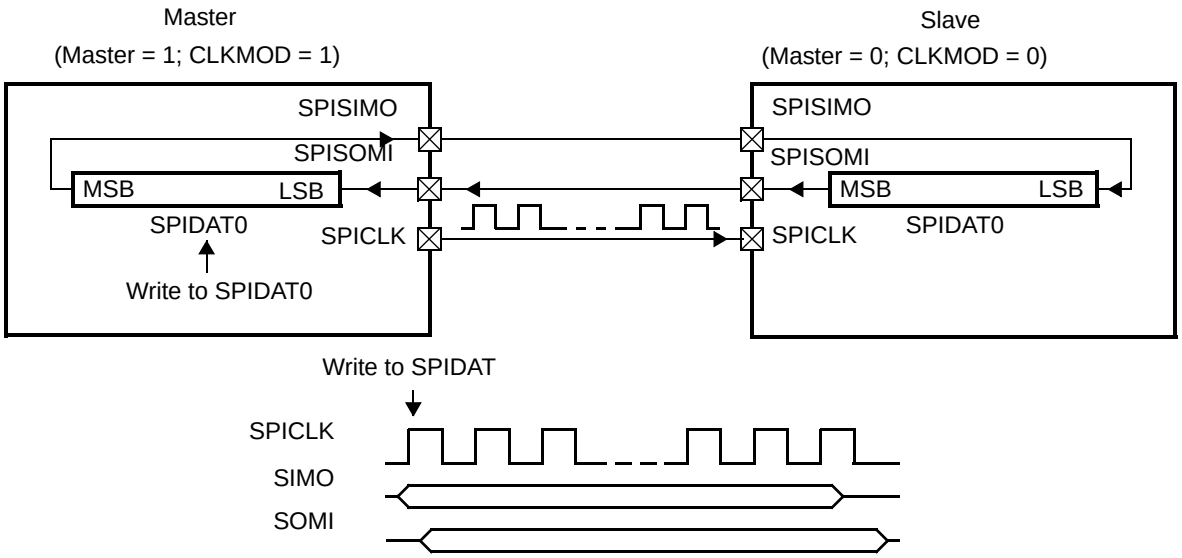
Data written to the shift register (SPIDAT0 / SPIDAT1) initiates data transmission on the SPISIMO pin, MSB first. Simultaneously, received data is shifted through the SPISOMI pin into the LSB of the SPIDAT0 register. When the selected number of bits have been transmitted, the received data in the shift register is transferred to the SPIBUF register for the CPU to read. Data is stored right-justified in SPIBUF.

See Section 11.2.2.1, *Data Sequencing when SPIDAT0 or SPIDAT1 Is Written* on page 449 and Section 11.2.2.2, *Data Sequencing when All Bits Shifted into RXSHIFT Register* on page 449 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 SPIDAT1register before the beginning of the SPICLK signal.

Figure 11-2. SPI Three-Pin Operation



11.2.3 Operation with $\overline{\text{SPISCS}}$

In master mode, each chip select signal is used to select a specific slave. In slave mode, chip select signal is used to enable/disable the transfer. Chip-select functionality is enabled by setting one of the $\overline{\text{SPISCS}}$ [7:0] pins as chip selects. It is disabled by setting all $\overline{\text{SPISCS}}$ [7:0] pins as GIOs in SPIPC0[7:0]

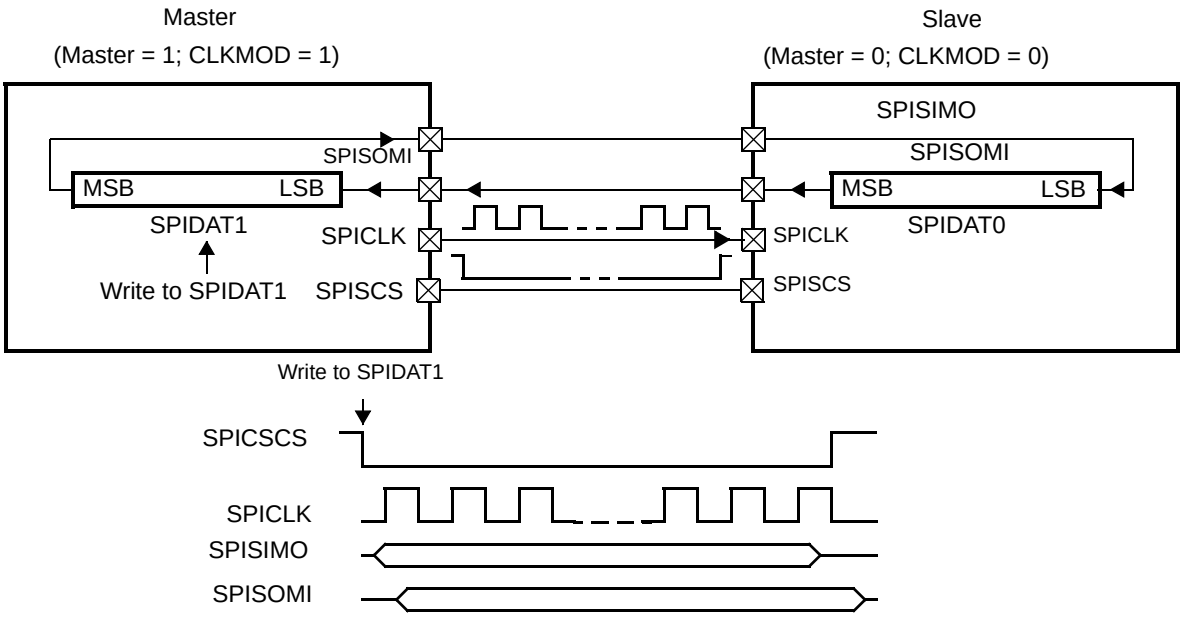
Multiple Chip Selects

The $\overline{\text{SPISCS}}$ [7:0] pins that are used must be configured as functional pins in the SPIPC0[7:0] register. The default pattern to be put on the $\overline{\text{SPISCS}}$ [7:0] 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 SPISCS[7:0] output pin. The drive state for SPISCS[7:0] 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 SPISCS input pin.

Figure 11-3. Operation with SPISCS



11.2.4 Operation with SPIENA

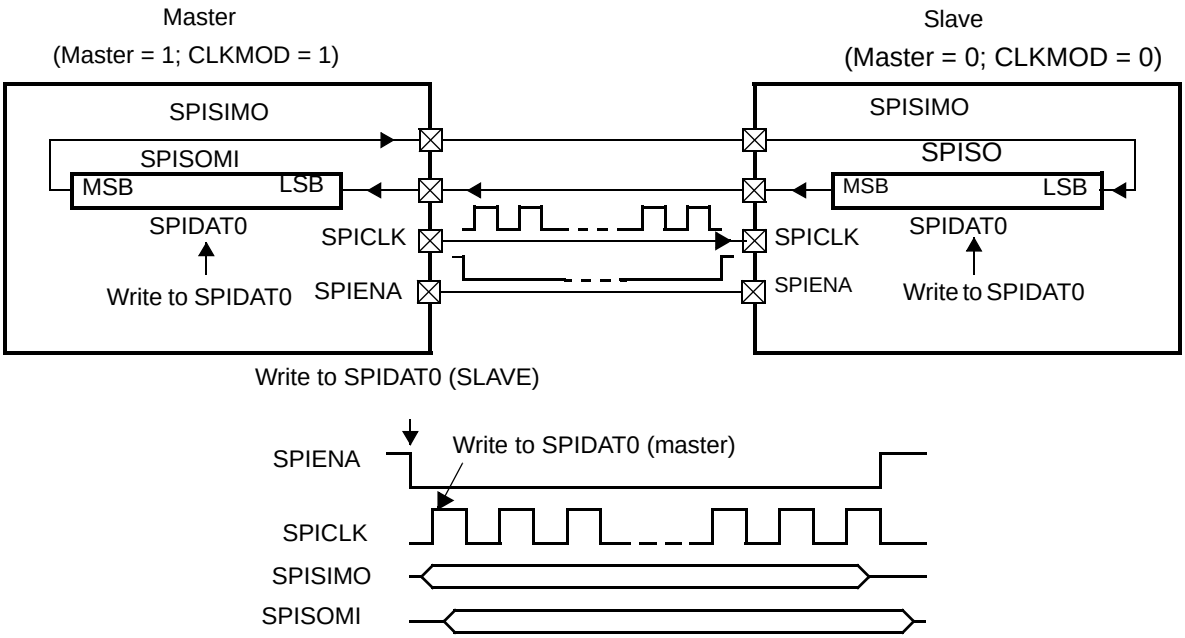
The SPIENA operates as a WAIT signal pin. For both the slave and the master, the SPIENA pin must be configured to be functional (SPIPC0[8] = 1). In this mode, an active low signal from the slave on the SPIENA pin allows the master SPI to drive the clock pulse stream. A high signal tells the master to hold the clock signal (and delay SPI activity).

If the SPIENA pin is in high-z 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 will act 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-z 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 11-4. Operation with $\overline{\text{SPIENA}}$



11.2.5 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 $\overline{\text{SPIENA}}$ pin and the required number of $\overline{\text{SPISCS}}$ [7:0] pins must be configured as functional pins.

If the $\overline{\text{SPIENA}}$ pin is in high-z mode (ENABLE_HIGHZ = 1), the slave SPI will put this signal into the high-impedance state by default. The slave will drive the signal $\overline{\text{SPIENA}}$ low when new data is written to the slave shift register and the slave has been selected by the master ($\overline{\text{SPISCS}}$ is low).

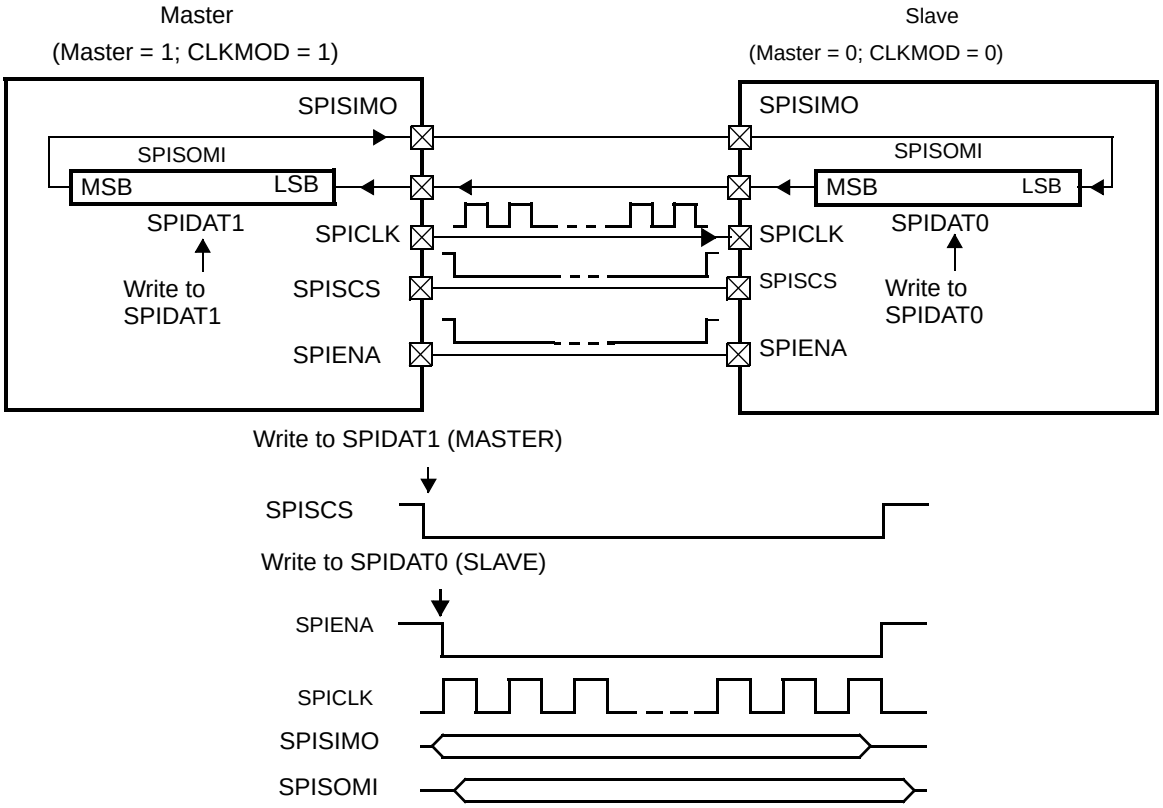
If the $\overline{\text{SPIENA}}$ pin is in push-pull mode (ENABLE_HIGHZ = 0), the slave SPI drives this pin high by default when it is in functional mode. The slave SPI will drive the $\overline{\text{SPIENA}}$ signal low when new data is written to the slave shift register (SPIDAT0/SPIDAT1) and the slave is selected by the master ($\overline{\text{SPISCS}}$ is low). If the slave is deselected by the master ($\overline{\text{SPISCS}}$ goes high), the slave $\overline{\text{SPIENA}}$ signal is driven high.

Note:
Push-pull mode of the $\overline{\text{SPIENA}}$ pin can be used only when there is a single slave in the system. When multiple SPI slave devices are connected to the common $\overline{\text{SPIENA}}$ pin, all of the slaves should configure their $\overline{\text{SPIENA}}$ pins in high-Z mode.

In master mode, if the $\overline{\text{SPISCS}}$ 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 $\overline{\text{SPISCS}}$ signals low. The master will drive the $\overline{\text{SPISCS}}$ signals high again after completing the transfer of the bits of the data.

In slave mode (CLKMOD = 0), the $\overline{\text{SPISCS}}$ pins will act as SPI functional inputs.

Figure 11-5. SPI Five-Pin Option with $\overline{\text{SPIENA}}$ and $\overline{\text{SPISCS}}$



11.2.6 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 11-6 shows how a 12-bit word (0xEC9) needs to be written to the transmit buffer to be transmitted correctly.

Figure 11-6. Format for Transmitting an 8-Bit Word

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
x	x	x	x	x	x	x	x	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 11-7 shows how a 10-bit word (0x0A2) is stored in the buffer once it is received.

Figure 11-7. Format for Receiving an 8-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

11.2.7 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.

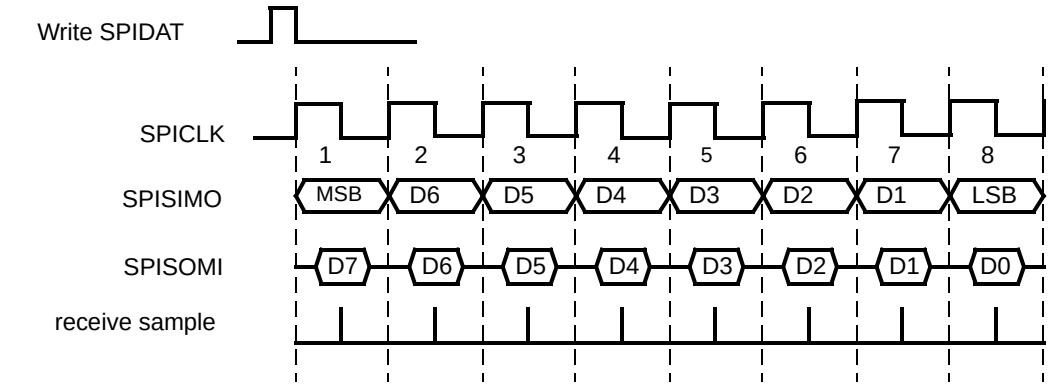
The data input and output edges depend on the values of both POLARITY and PHASE as shown in Table 11-2.

Table 11-2. 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.

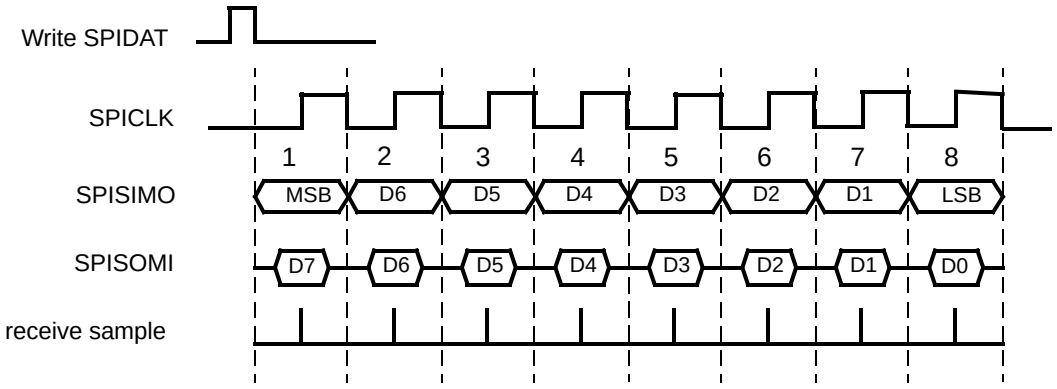
Figure 11-8 to Figure 11-11 illustrate the four possible configurations of SPICLK corresponding to each mode. Having four signal options allows the SPI to interface with many different types of serial devices.

Figure 11-8. Clock Mode with Polarity = 0 and Phase = 0



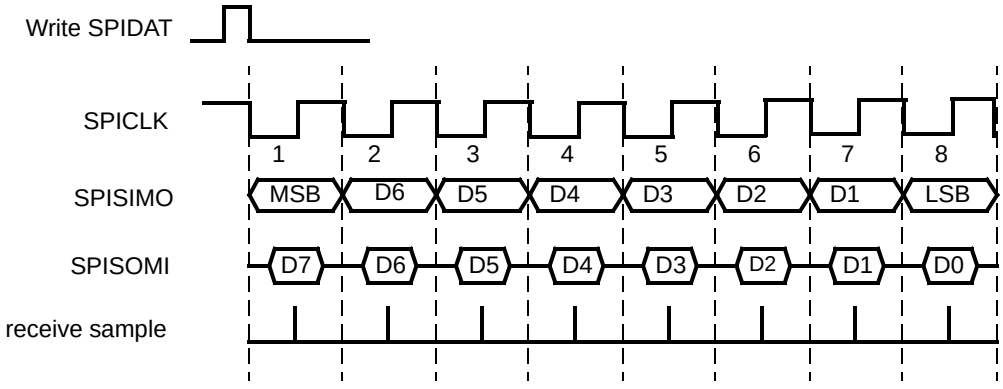
Data is output on the rising edge of SPICLK.
Input data is latched on the falling edge of SPICLK.

Figure 11-9. Clock Mode with Polarity = 0 and Phase = 1



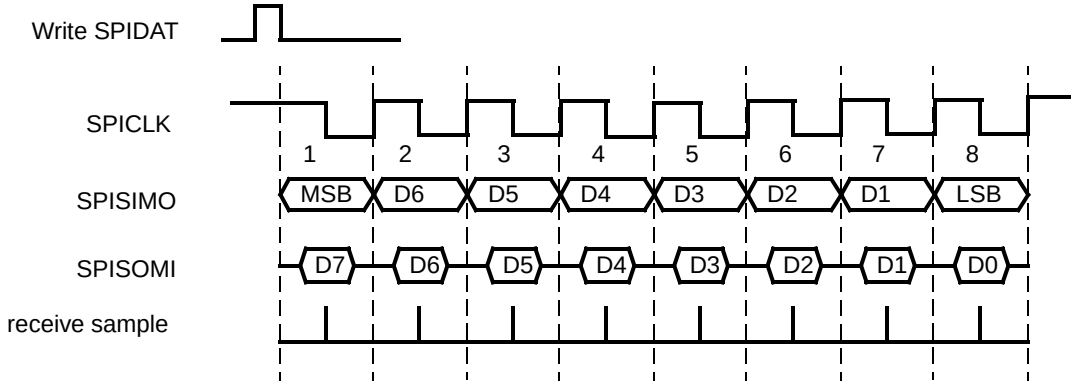
Data is output one-half cycle before the first rising edge of SPICLK and on subsequent falling edges of SPICLK
Input data is latched on the rising edge of SPICLK

Figure 11-10. 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 11-11. Clock Mode with Polarity = 1 and Phase = 1

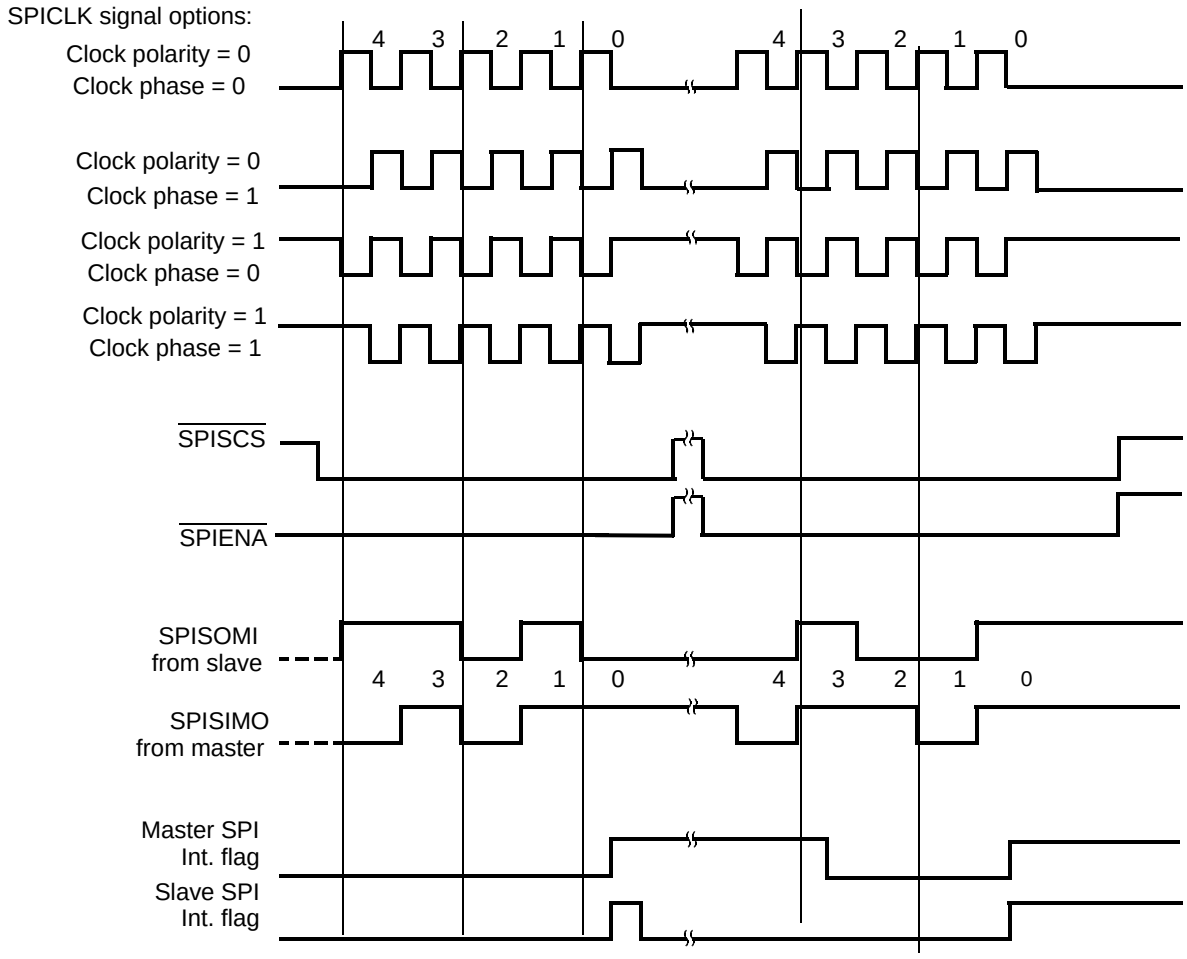


Data is output one-half cycle before the first falling edge of SPICLK and on the subsequent rising edges of SPICLK. Input data is latched on the falling edge of SPICLK.

11.2.8 Data Transfer Example

Figure 11-12 illustrates a SPI data transfer between two devices using a character length of five bits.

Figure 11-12. Five Bits per Character (5-Pin Option)



11.2.9 Decoded and Encoded Chip Select (Master Only)

In this device the SPI can connect to up to 4 individual slave devices using chip-selects by routing one wire to each slave. The 4 chip selects in the control field are directly connected to the 4 pins. The default value of each chip select, i.e., not active, can be configured via the register CSDEF. During a transmission, the value of the chip select control field (CSNR[7:0]) of the SPIDAT1 register (SPIDAT1[23:16]) is driven on the SPISCS [7:0] pins. When the transmission finishes the default chip-select value (defined by the CSDEF register) is put on the SPISCS [7:0] pins.

The SPI can support more than 4 slaves by using encoded chip selects. To connect the SPI with encoded slaves devices, the CSNR field allows multiple active SPISCS pins at the same time, which enables binary encoded chip selects from 0 to 16. To use encoded chip selects, all four 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* SPISCS pins can be used for encoding a *n*-bit address and the remaining pins can be connected to decoded-mode slaves.

11.2.10 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 peripheral clock(VCLK).

If a particular data format specifically does not require these additional set-up or hold times for the chip select pin(s), then they can be disabled in the corresponding SPIFMTx register.

11.2.11 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.

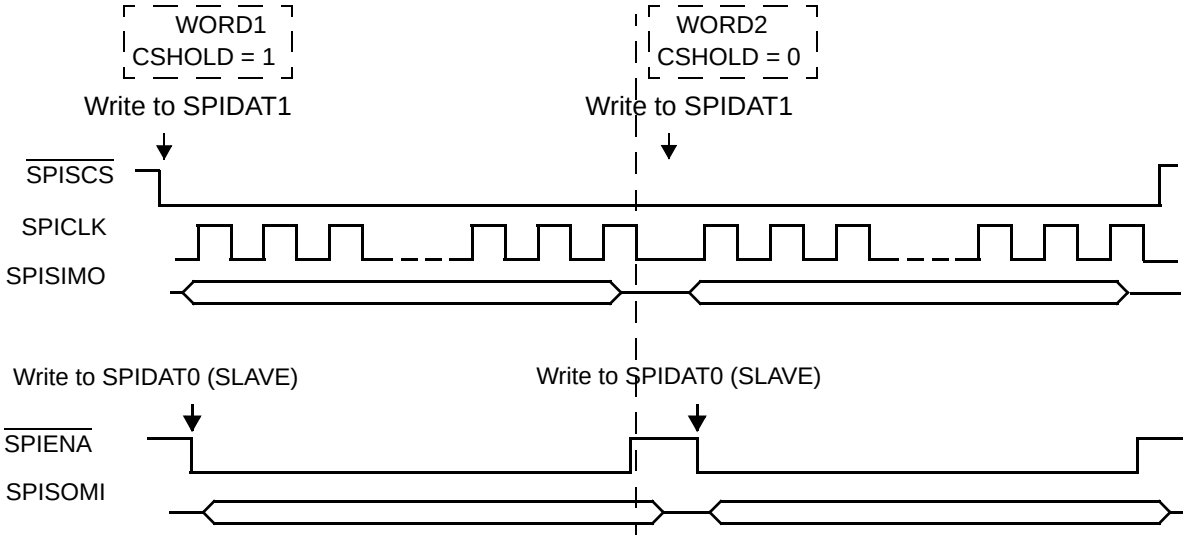
11.2.11.1 The CSHOLD Bit in Master Mode

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 11-13 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 11-13. Typical Diagram when a Buffer in Master is in CSHOLD Mode (SPI-SPI)



11.2.12 Detection of Slave Desynchronization (Master Only)

When a slave supports generation of an enable signal (ENA), de-synchronization can be detected. With the enable signal a slave indicates to the master that it is ready to exchange data. A de-synchronization 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 de-synchronized slave and the consecutive data word. A configurable 8 bit time-out counter, 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 Desync 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 guaranteed to be for the current buffer.

11.2.13 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. 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.

11.2.14 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 (e.g. 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 $\overline{\text{SPISCS}}$ 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 $\overline{\text{SPISCS}}$(x) pins are configured as functional, and are being used for selecting the slave.

11.2.15 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 MIBSPIP supports Parellel 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[1:0] bits of SPIPMCTRL register. See Figure 11-28. for details about this register.

After reset the parallel mode selection bits are cleared (single SIMO/SOMI lines).

11.2.15.2 Parallel Mode Block Diagram

Figure 11-14 and Figure 11-15 show the parallel connections to the SPI shift register.

Figure 11-14. Block Diagram Shift Register, MSB First

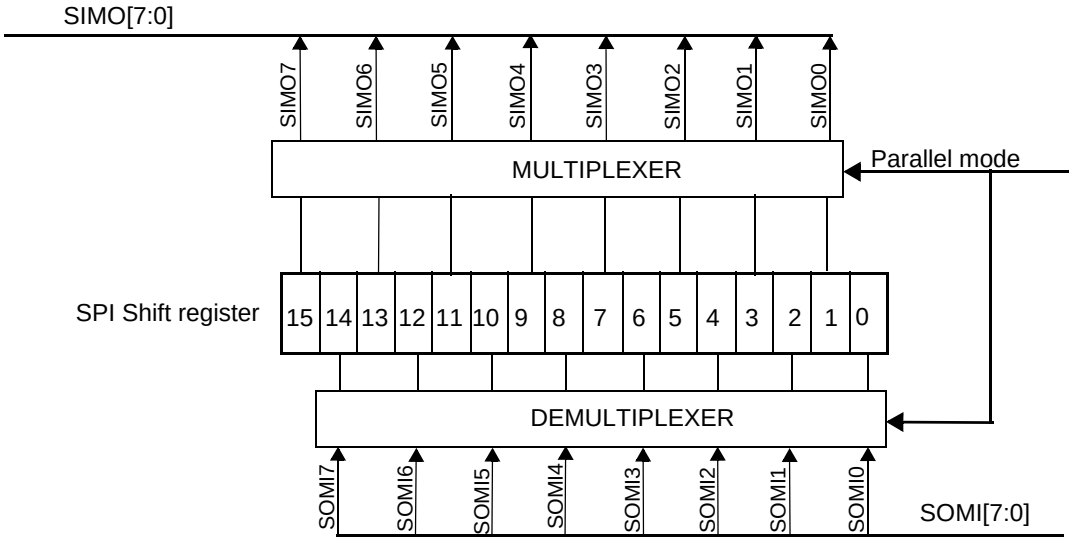
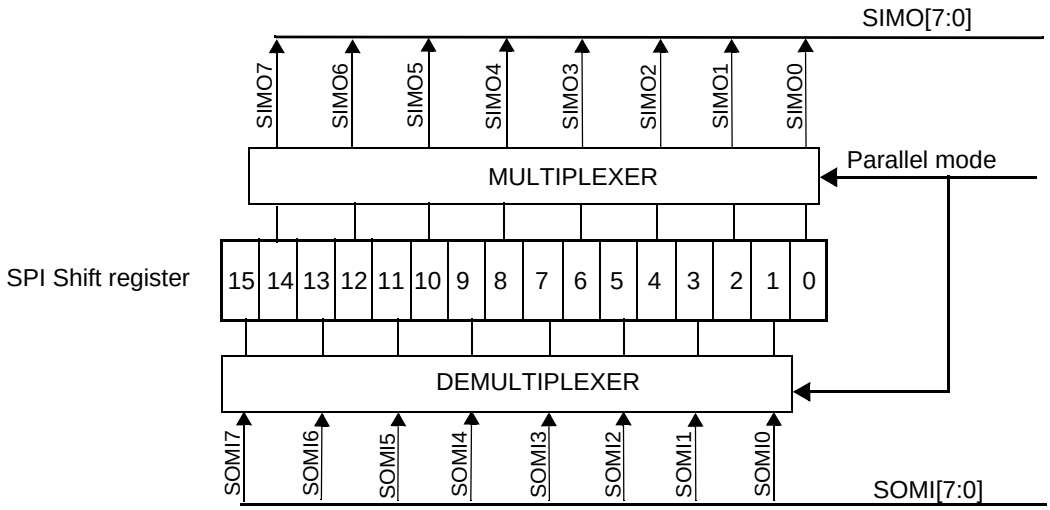


Figure 11-15. Block Diagram Shift Register, LSB First



11.2.15.3 Parallel Mode Pin Mapping, MSB First

Table 11-3 and Table 11-4 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 11-3. 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 11-4. 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

11.2.15.4 Parallel Mode Pin Mapping, MSB-First, LSB-First

Table 11-5–Table 11-6 describe the SIMO and SOMI pin mapping when SPI is used in parallel mode (1, 2, 4, 8) pin mode, LSB first.

Table 11-5. 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 11-6. 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

11.2.15.5 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 11-16 shows the clock /data diagram of the 2-data line mode. Figure 11-17 shows the timing of a two-pin parallel transfer.

Figure 11-16. 2-data Line Mode (Phase 0, Polarity 0)

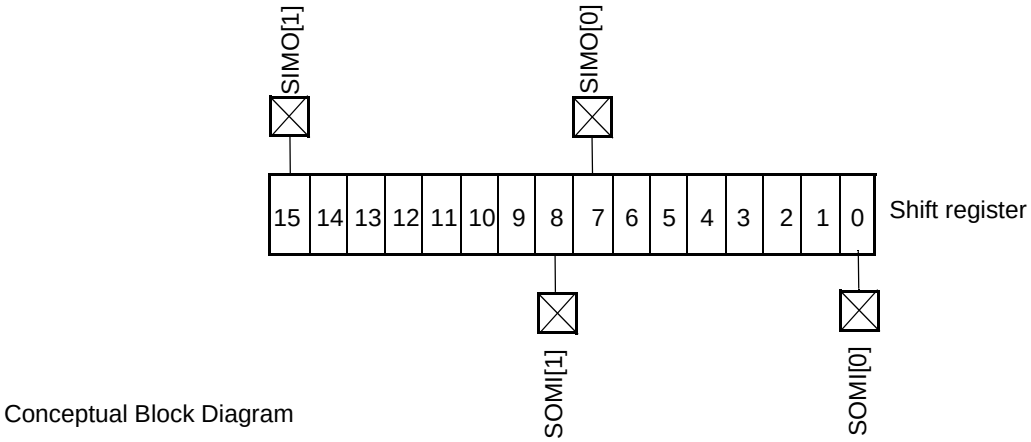
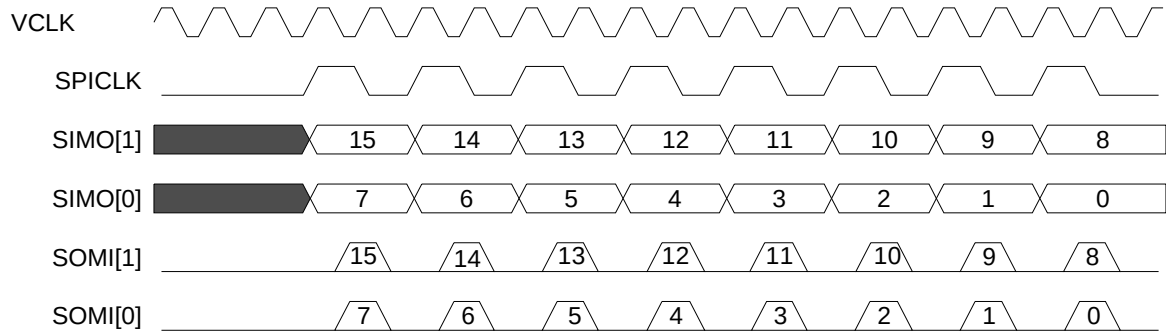


Figure 11-17. Two-Pin Parallel Mode Timing Diagram (Phase 0, Polarity 0)



11.2.15.6 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 11-19 shows the clock/data diagram of the four-data line mode. Figure 11-20, shows the timing diagram for four-data line mode.

Figure 11-18. 4-Data Line Mode (Phase 0, Polarity 0)

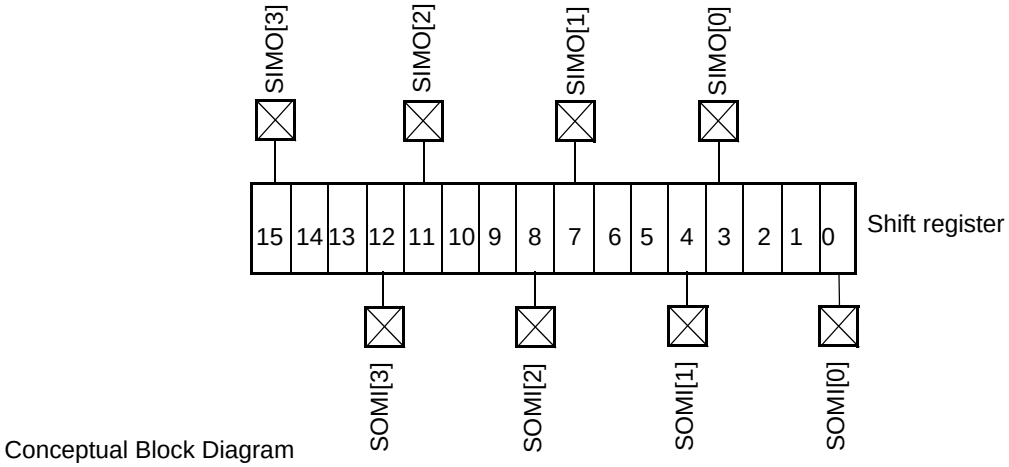


Figure 11-19. 4-Data Line Mode (Phase 0, Polarity 0)

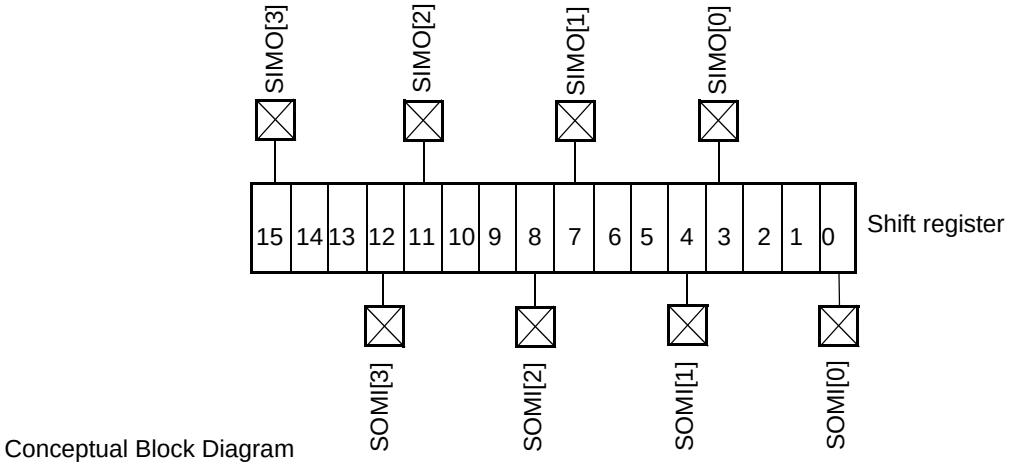
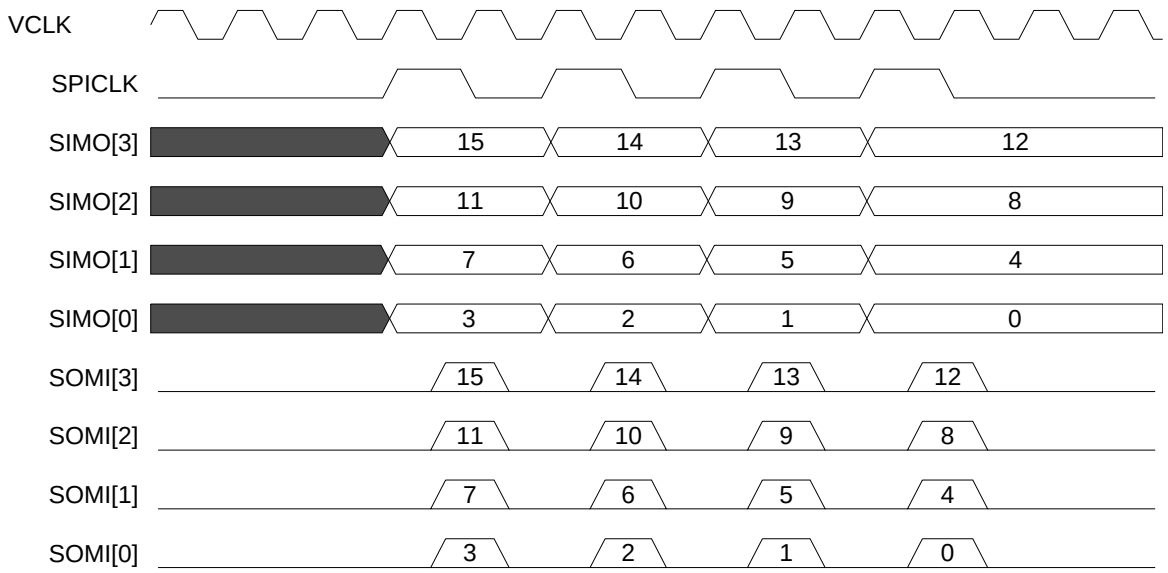


Figure 11-20. 4 Pins Parallel Mode Timing Diagram (Phase 0, Polarity 0)



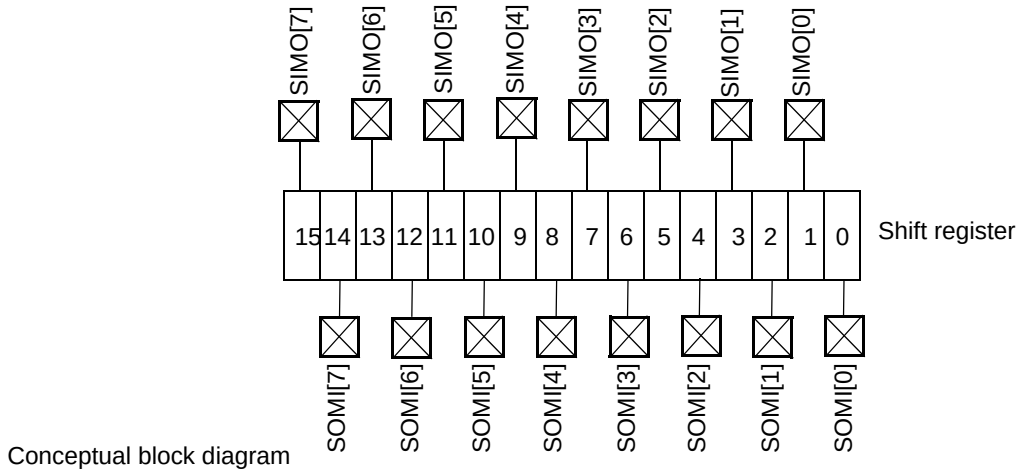
11.2.15.7 8-Data Line Mode (MSB First, Phase 0, Polarity 0)

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 one 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 11-21 shows the clock/data diagram of the 8-data line mode. Figure 11-22 shows the pin timings for 8-data line mode.

Figure 11-21. eight-data Line Mode (Phase 0, Polarity 0)

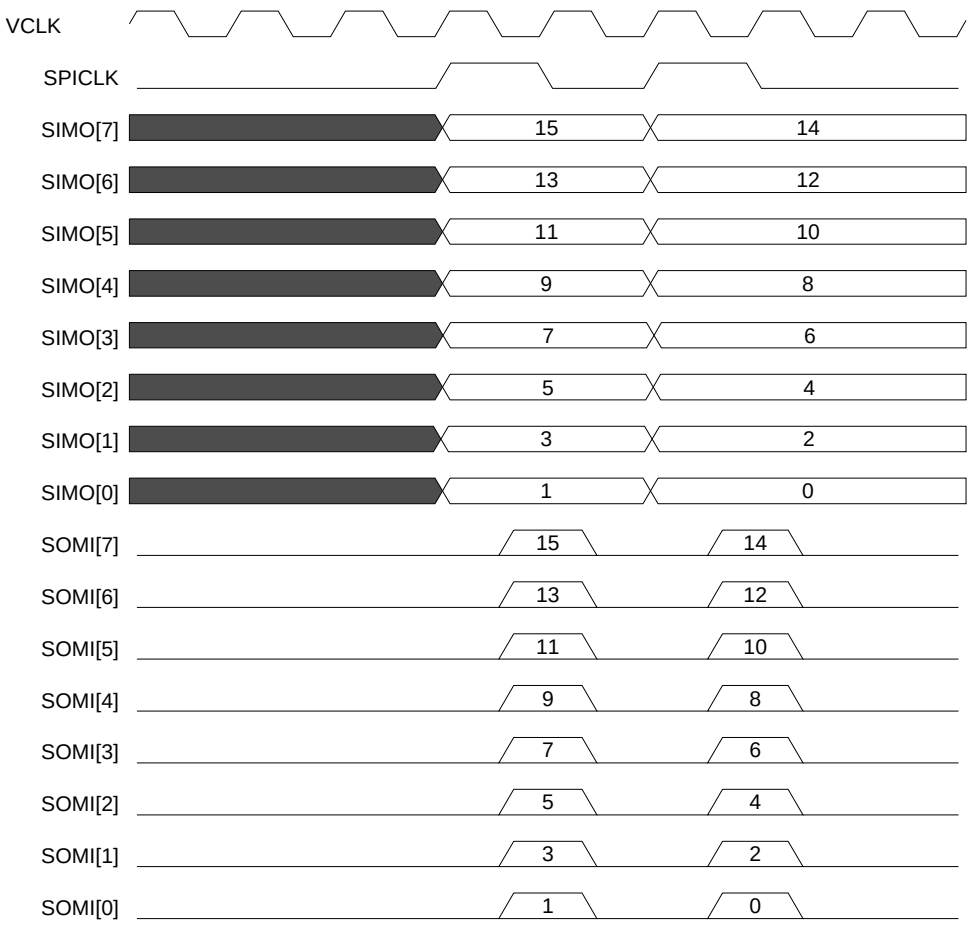


Conceptual block diagram

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 11-22. 8 Pins Parallel Mode Timing Diagram (Phase 0, Polarity 0)



Note: Modulo Count Parallel Mode is not supported in this device.

11.2.16 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.

11.3 Test Features

11.3.1 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 fedback to the receiver, whereas the SIMO, SOMI, and CLK pin are disconnected, i.e., 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.

11.3.2 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 11-23 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 11.9.42, *I/O-Loopback Test Control Register (IOLPBKTSTCR)* on page 566 for details of the register field controlling these features.

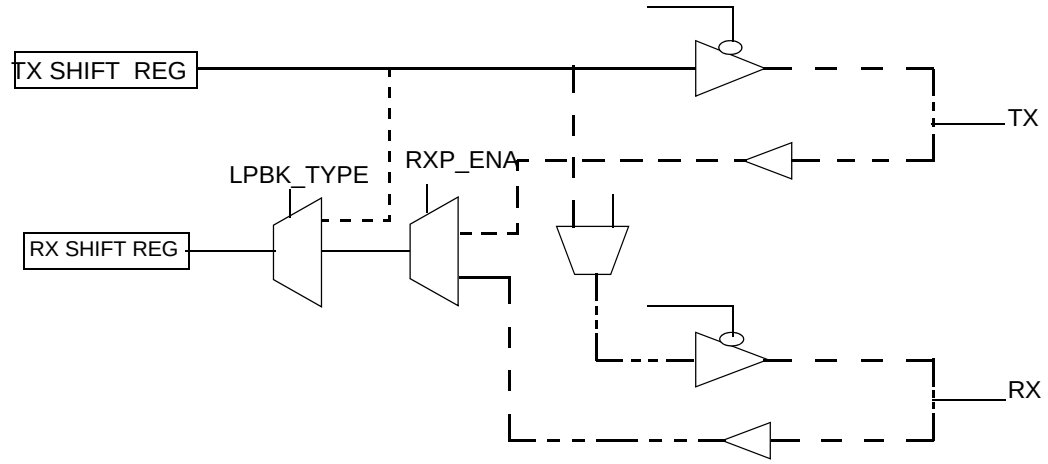
In Input/Output loopback test modes, even when the module is in slave mode, the SPICLK is generated internally. This SPICLK is used for all loopback-mode SPI transactions. Slave-mode features can be tested without the help of another master SPI, using the internally-generated SPICLK. Chip selects are 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 nENA 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 11-23. I/O Paths during I/O Loopback Modes



1 This diagram is intended to illustrate loopback paths and therefore may omit some normal-mode paths.

11.3.2.1 IO 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 SPISCS[3:0] to 0x0. The actual number of chip selects can be programmed to have any or all of the SPISCS 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[3:0] 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.

11.4 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
- input-buffer enabling/disabling (controlled by the PULDIS and PSEL bits)

11.5 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.

11.6 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.

11.6.1 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:

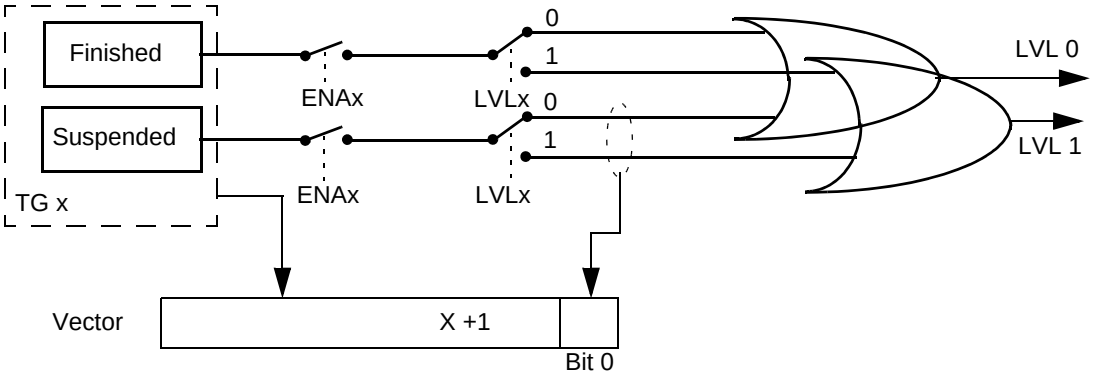
- Transmission error interrupt
- Receive overrun interrupt
- TG suspended interrupt
- TG completed interrupt

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 11-24 illustrates the TG interrupts.

Figure 11-24. TG Interrupt Structure

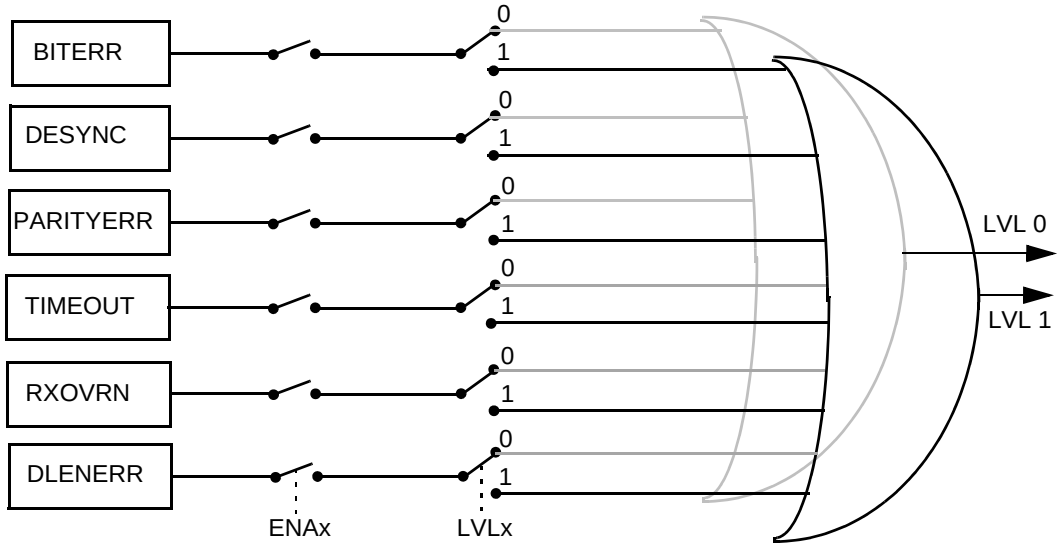


During transmission, if one of the following errors occurs, BITERR, DESYNC, PARITYERR, TIMEOUT, DLENERR, the corresponding flag in the SPIFLG register is set. If the enable bit is set, then an interrupt is generated. The level of the interrupts could be generated according to the bit field in SPILVL 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 11-25. 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.

11.7 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:

- DMA REQ EN (SPIINT0[16]) is set to 1 while SPIEN (SPIGCR1[24]) is already 1.
- SPIEN (SPIGCR1[24]) is set to 1 while DMA REQ EN (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.

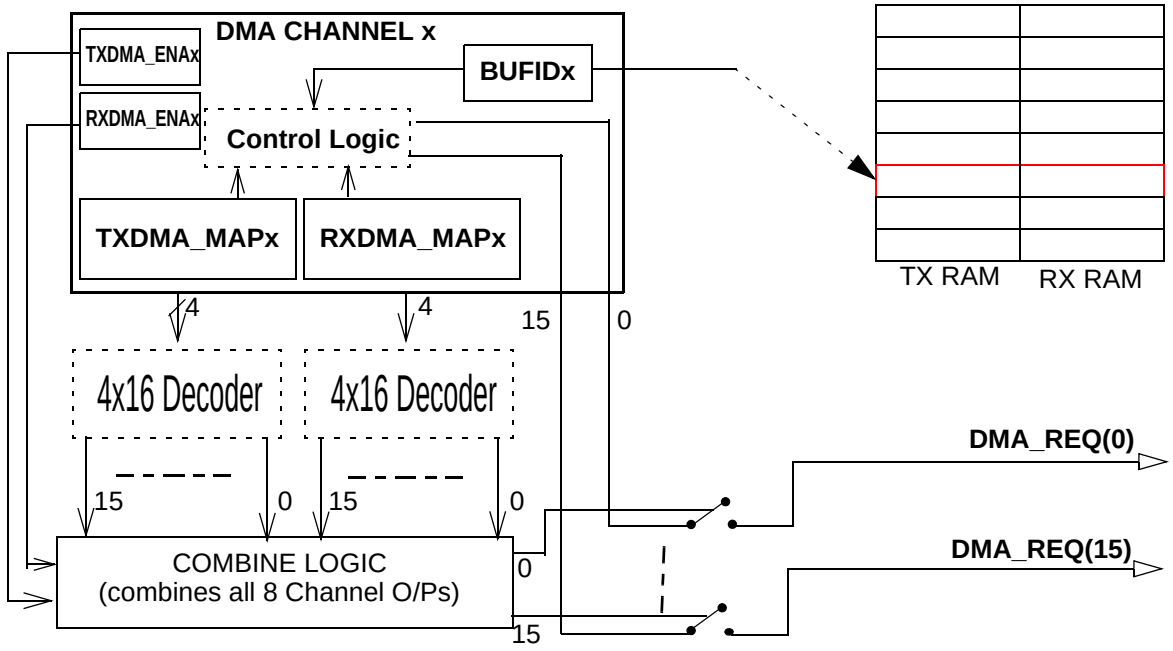
11.7.1 DMA in Multi-Buffer Mode

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 12. DMA Channel and Request Line (logical) structure in Multibuffer Mode



11.8 Module Configuration

MibSPI/MibSPIP can be configured to function as Normal SPI and Multibuffered 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.

11.8.1 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 SPIENA is held low 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, CLK and optional CSx, ENA 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 CS 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 SPIENA to 1 after the SPI is configured.

– Perform Transmit and receive data, using SPIDAT1 and SPIBUF register.

– User must wait for TXFULL to reset or TXINT before writting next data to SPIDAT1 register.

– User must wait for RXEMPTY to reset or RXINT before reading the data from SPIBUF register.

11.8.2 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 SPIENA is held low 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 to 1 to get access to multi-buffer mode registers.

– Configure the SIMO, SOMI, CLK and optional CSx, ENA 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 SPIENA 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 16 Bit Control fields in every TXRAM buffer in the TG has to be configured.

– Configure one of the eight BUFMODE available for each buffer.

– Fill the datas 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 occurence 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.

– User can poll Transfer-group interrupt flag or wait for a transfer-completed interrupt to read and write new data to the buffers.

11.9 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 can be found in the device data sheet.

Table 11-7. SPI Registers

Offset Address[1] 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
0x00h SPIGCR0 page 483	Reserved															
	Reserved															nRESET
0x04h SPIGCR1 page 484	Reserved							SPIEN	Reserved							LOOP BACK
	Reserved							POWER DOWN	Reserved						CLK MOD	MAS-TER
0x08h SPIINT0 page 486	Reserved							ENABLE HIGH Z	Reserved							DMA REQ EN
	Reserved						TXINT ENA	RXINT ENA	Reserved	OVRN INT ENA	Reserved	BIT ERR ENA	DESYNC ENA	PAR ERR ENA	TIME OUT ENA	DLEN ERR ENA
0x0Ch SPILVL page 489	Reserved															
	Reserved						TX INT LVL	RX INT LVL	Reserved	OVRN INT LVL	Reserved	BITERR LVL	DESYNC LVL	PAR ERR LVL	TIME OUT LVL	DLEN ERR LVL
0x10h SPIFLG page 491	Reserved							BUF INIT ACTIVE	Reserved							
	Reserved						TX INT FLG	RX INT FLG	Reserved	OVRN INT FLG	Reserved	BIT ERR FLG	DESYNC FLG	PAR ERR FLG	TIME OUT FLG	DLEN ERR FLG
0x14h SPIPC0 page 496	SOMIFUN[7:0]								SIMOFUN[7:0]							
	Reserved					SOMI FUN	SIMO FUN	CLK FUN	ENA FUN	SCSFUN[7:0]						
0x18h SPIPC1 page 498	SOMIDIR[7:0]								SIMODIR[7:0]							
	Reserved					SOMI DIR	SIMO DIR	CLK DIR	ENA DIR	SCSDIR[7:0]						

1 The base address of these registers can be found in the device data sheet.

Offset Address[1] 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
0x1Ch SPIPC2 page 500	SOMIDIN[7:0]								SIMODIN[7:0]							
	Reserved				SOMI DIN	SIMO DIN	CLK DIN	ENA DIN	SCSDIN[7:0]							
0x20h SPIPC3 page 502	SOMIDOUT[7:0]								SIMODOUT[7:0]							
	Reserved				SOMI DOUT	SIMO DOUT	CLK DOUT	ENA DOUT	SCSDOUT[7:0]							
0x24h SPIPC4 page 504	SOMISET[7:0]								SIMOSET[7:0]							
	Reserved				SOMI SET	SIMO SET	CLK SET	ENA-SET	SCSSET[7:0]							
0x28h SPIPC5 page 506	SOMICLR[7:0]								SIMOCLR[7:0]							
	Reserved				SOMI CLR	SIMO CLR	CLK CLR	ENA CLR	SCSCLR[7:0]							
0x2Ch SPIPC6 page 508	SOMIPDR[7:0]								SOMICLR[7:0]							
	Reserved				SOMI PDR	SIMO PDR	CLK PDR	ENA PDR	SCSPDR[7:0]							
0x30h SPIPC7 page 510	SOMIDIS[7:0]								SIMODIS[7:0]							
	Reserved				SOMI PDIS	SIMO PDIS	CLK PDIS	ENA PDIS	SCSPDIS[7:0]							
0x34h SPIPC8 page 512	SOMIPSEL[7:0]								SIMOPSEL[7:0]							
	Reserved				SOMI PSL	SIMO PSL	CLK PSL	ENA PSL	SCSPSL[7:0]							
0x38h SPIDAT0 page 514	Reserved															
	TXDATA(15–0)															

1 The base address of these registers can be found in the device data sheet.

Offset Address[1] 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
0x3Ch SPIDAT1 page 515	Reserved			CS HOLD	Reserved	WDEL	DFSEL		CSNR(7–0)							
	TXDATA(15–0)															
0x40h SPIBUF page 517	RXEMPTY	RX OVR	TX FULL	BIT ERR	DE SYNC	PARITYERR	TIME OUT	DLEN ERR	LCSNR(7–0)							
	RXDATA(15–0)															
0x44h SPIEMU page 521	Reserved															
	RXDATA(15–0)															
0x48h SPIDELAY page 522	C2TDELAY(7–0)								T2CDELAY(7–0)							
	T2EDELAY(7–0)								C2EDELAY(7–0)							
0x4Ch SPIDEF page 526	Reserved															
	Reserved								CSDEF[7:0]							
0x50h–5Ch SPIFMT[0:3] page 527	Reserved		WDELAY[0:3](5–0)						PAR POL [0:3]	PAR-ITY [0:3] ENA	WAIT ENA [0:3]	SHIFT DIR[0:3]	Reserved	DIS CS TIM-ERS	POLARITY [0:3]	PHASE [0:3]
	PRESCALE[0:3]								Reserved			CHARLEN[0:3]				
0x60h–64h TGINTVECT[0:1] page 530	Reserved															
	Reserved										INTVECT[0:1]					SUS PEND [0:1]
0x6Ch SPIPMCTRL page 534	Reserved		MOD CLK POL3	MMODE 3(2–0)			PMODE 3(1–0)		Reserved		MOD CLK POL2	MMODE 2(2–0)			PMODE 2(1–0)	
	Reserved		MOD CLK POL1	MMODE 1(2–0)			PMODE 1(1–0)		Reserved		MOD CLK POL0	MMODE0(2–0)			PMODE 0(1–0)	

1 The base address of these registers can be found in the device data sheet.

Offset Address[1] 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
0x70h MIBSPIE page 538	Reserved															RX RAM ACCESS
	Reserved															MibSPI ENA
0x74h TGITENST page 540	SET INTEN RDY[15:0]															
	SET INTEN SUS[15:0]															
0x78h TGITENCR page 541	CLR INTEN RDY[15:0]															
	CLR INTEN SUS[15:0]															
0x7Ch TGITLVST page 542	SET INTLVL RDY[15:0]															
	SET INTLVL SUS[15:0]															
0x80h TGITLVCR page 543	CLR INTLVL RDY[15:0]															
	CLR INTLVL SUS[15:0]															
0x84h TGITFLG page 544	INTFLG RDY[15:0]															
	INTFLG SUS[15:0]															
0x88–0x8Ch	Reserved															
	Reserved															
0x90h TICKCNT page 546	TICK ENA	RE LOAD	CLKCTRL(1–0)	Reserved												
	TICKVALUE(15–0)															

1 The base address of these registers can be found in the device data sheet.

Offset Address[1] 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
0x94h LTGPEND page 548	Reserved			TG IN SERVICE(4–0)					Reserved							
	Reserved	LPEND(6–0)							Reserved							
0x98–D4h TG[0:15]CTRL page 549	TG ENA[0:15]	ONE SHOT [0:15]	PRST [0:15]	TGTD[0:15]	Reserved				TRGEVT[0:15](4–0)				TRIGSRC[0:15](4–0)			
	Reserved	PSTART[0:15](6–0)							Reserved	PCURRENT[0:15](6–0)						
0xD8h–F4h DMA[0:8]CTRL page 555	ONE SHOT [0:8]	BUFID[0:8](6–0)							RXDMA_MAP[0:8](3–0)				TXDMA_MAP(3–0)			
	RX DMA ENA[0:8]	TX DMA ENA[0:8]	NO BRK[0:8]	ICOUNT[0:8](4–0)					Reserved	COUNTBIT17[0:8]	COUNT[0:8](5–0)					
0xF8 - 0x114h DMAxCOUNT page 559	ICOUNTx(15–0)															
	COUNTx(15–0)															
0x118h DMACNTLEN page 560	Reserved															
	Reserved															LARGE COUNT
0x11Ch	Reserved															
	Reserved															
0x120h UERRCTRL page 561	Reserved															
	Reserved							PTEST-EN	Reserved				EDEN(3–0)			
0x124h UERRSTAT page 562	Reserved															
	Reserved														EDFLG1	EDFLG0

1 The base address of these registers can be found in the device data sheet.

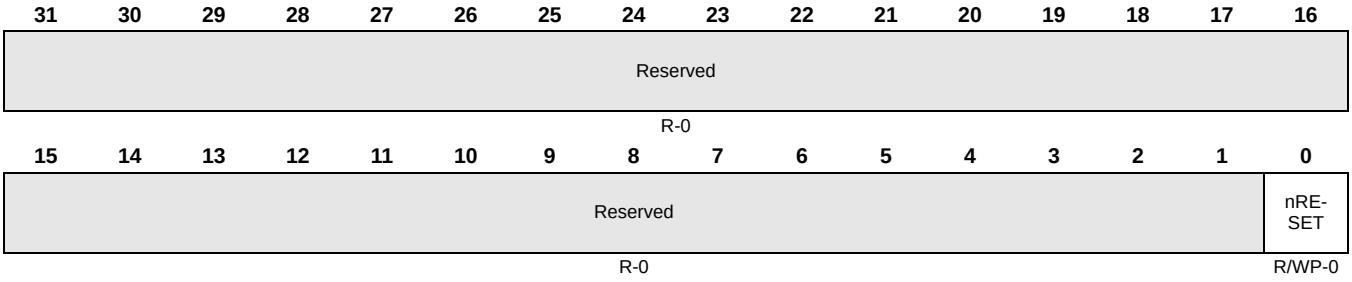
Offset Address[1] Register	31 15	30 14	29 13	28 12	27 11	26 10	25 9	24 8	23 7	22 6	21 5	20 4	19 3	18 2	17 1	16 0
0x128h UERRADDR1 page 563	Reserved															
	Reserved							UERRADDR1(9–0)								
0x12Ch UERRADDR0 page 564	Reserved															
	Reserved								UERRADDR0(8–0)							
0x130h RXOVRN_BUF_ ADDR page 565	Reserved															
	Reserved							RXOVRN_BUF_ADDR(9–0)								
0x134h IOLPBKTSTCR page 566	Reserved						SCS FAIL FLG	Reserved			CTRL_ BITERR	CTRL_ DESYNC	CTRL_ PAR ERR	CTRL_ TIME OUT	CTRL_ DLEN ERR	
	Reserved			IOLPBKTSTENA(3–0)				Reserved		ERR SCS PIN(2–0			CTRL SCS PIN ERR	LPBK_ TYPE	RXP_ ENA	

1 The base address of these registers can be found in the device data sheet.

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.

11.9.1 SPI Global Control Register 0 (SPIGCR0)

Figure 11-1. SPI Global Control Register 0 (SPIGCR0) [offset = 00h]



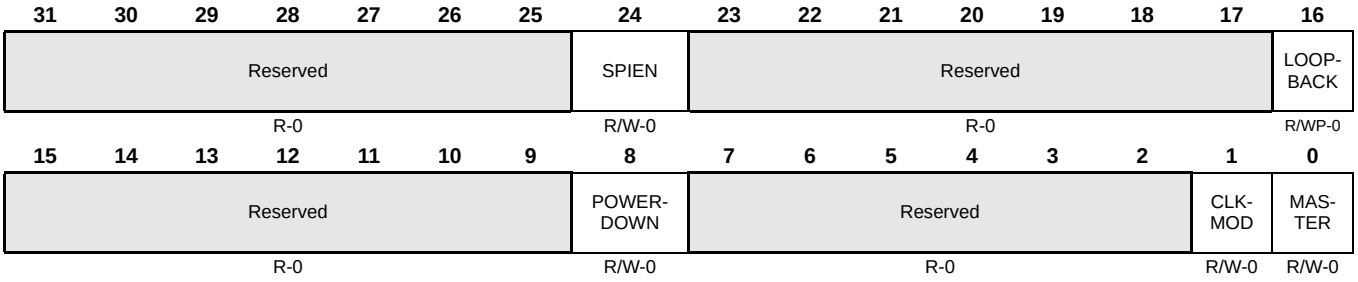
R = Read in all modes; WP = Write in privilege mode only; -n = value after reset

Table 11-8. SPI Global Control Register 0 (SPIGCR0) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved		Reads return zero and writes have no effect.
0	nRESET		Reset bit for the module. This bit needs to be set to 1 before any operation on SPI can be done. After setting this bit, auto Initialization of multi-buffer RAM starts.
		0	SPI is in the reset state.
		1	SPI is out of the reset state.

11.9.2 SPI Global Control Register 1 (SPIGCR1)

Figure 11-2. SPI Global Control Register 1 (SPIGCR1) [offset = 04h]



R = Read in all modes; W = Write in all modes; WP = Write in privilege mode only; -n = value after reset

Table 11-9. SPI Global Control Register 1 (SPIGCR1) Field Descriptions

Bit	Name	Value	Description
31–25	Reserved		Reads return zero and 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 SPIDAT0 and SPIDAT1 registers •All the fields of the SPIFLG register •Contents of SPIBUF and the internal RXBUF registers
		0	The SPI is not activated for transfers.
		1	Activates SPI
23–17	Reserved		Reads return zero and writes have no effect.
16	LOOPBACK		Internal loop-back test mode. The internal self-test option can be enabled by setting this bit. If the SPISIMO and SPISOMI pins are configured with SPI functionality, then the SPISIMO[7:0] pins areis 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 disabled.
		1	Internal loop-back test mode enabled.

Table 11-9. SPI Global Control Register 1 (SPIGCR1) Field Descriptions (Continued)

Bit	Name	Value	Description
15–9	Reserved		Reads return zero and writes have no effect.
8	POWERDOWN		When active, the SPI state machine enters a power-down state.
		0	The SPI is in active mode.
		1	The SPI is in power-down mode.
7–2	Reserved		Reads return zero and 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 SPIENA and SPISCS[3:0] pins in functional mode.
		0	Clock is external. •SPIENA is an output. •SPISCS[3:0] are inputs.
		1	Clock is internally-generated. •SPIENA is an output. •SPISCS[3:0] are outputs.
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 set 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 is anare inputs, SPISOMI[7:0] pins is anare outputs
		1	SPISOMI[7:0] pins is anare inputs, SPISIMO[7:0] pins is anare outputs

11.9.3 SPI Interrupt Register (SPIINT0)

Figure 11-3. SPI Interrupt Register (SPIINT0) [offset = 08h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved							ENABLE-HIGHZ	Reserved							DMAREQEN
R-0							R/W-0	R-0							R/W-0
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved						TXINTENA	RXINT-ENA	Reserved	RXOVRNIN-TENA	Reserved	BITER-RENA	DESYNCENA	PAR-ERRENA	TIME-OUT-ENA	DLEN-ERRENA
R-0						R/W-0	R/W-0	R-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

R = Read, W = write, P = Privilege mode; -*n* = Value after reset

Table 11-10. SPI Interrupt Register (SPIINT0) Field Descriptions

Bit	Name	Value	Description
31–25	Reserved		Reads return zero and 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		Reads return zero and writes have no effect.
16	DMAREQEN		DMA request enable. Enables the DMA request signal to be generated for both receive and transmit channels. 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		Reads return zero and writes have no effect.
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 (SPIFLG[9]) is set to 1.
		0	No interrupt will be generated upon TXINTFLG being set to 1.

Table 11-10. SPI Interrupt Register (SPIINT0) Field Descriptions (Continued)

Bit	Name	Value	Description
		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 (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		Reads return zero and writes have no effect.
6	RXOVRNINTENA		Overrun interrupt enable.
		0	Overrun interrupt will not be generated.
		1	Overrun interrupt will be generated.
5	Reserved		Reads return zero and writes have no effect.
4	BITERRENA		Enables interrupt on bit error.
		0	No interrupt asserted upon bit error.
		1	Enables an interrupt on a bit error.
3	DESYNCENA		Enables interrupt on desynchronized slave. DESYNCENA is used in master mode only.
		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.

Table 11-10. SPI Interrupt Register (SPIINT0) Field Descriptions (Continued)

Bit	Name	Value	Description
		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 SPIENA is used, if the 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 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 SPISCS pins are used, if the incoming valid SPISCS 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.

11.9.4 SPI Interrupt Level Register (SPILVL)

Figure 11-4. SPI Interrupt Level Register (SPILVL) [offset = 0Ch]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved						TXINTLVL	RX INTLVL	Reserved	RXOVRNIN-TLVL	Reserved	BITER-RLVL	DESYN-CLVL	PAR-ERRLVL	TIME-OUTLVL	DLEN-ERRLVL
R-0						R/W-0	R/W-0	R-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

R = Read, W = Write in any mode; -*n* = Value after reset

Table 11-11. SPI Interrupt Level Register (SPILVL) Field Descriptions

Bit	Name	Value	Description
31–10	Reserved		Reads return zero and 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		Reads return zero and 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		Reads return zero and 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.

Table 11-11. SPI Interrupt Level Register (SPILVL) Field Descriptions (Continued)

Bit	Name	Value	Description
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	DLEN ERR LVL		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.

11.9.5 SPI Flag Register (SPIFLG)

Note:
SPIFLAG bits are cleared on read. Software must check all flag bits when reading this register.

Figure 11-5. SPI Flag Register (SPIFLG) [offset = 10h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved							BUFINI-TAC-TIVE	Reserved							
R-0							R-0	R-0							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						TXINT-FLG	RXINT-FLG	Reserved	RXOVRNINT-FLG	Reserved	BITER-RFLG	DESYN-CFLG	PAR-ERRFLG	TIME-OUT-FLG	DLEN-ERRFLG
R-0						R-0	R/WC-0	R-0	R/WC-0	R-0	R/WC-0	R/WC-0	R/WC-0	R/WC-0	R/WC-0

R = Read; WC = Write/read Clear; -*n* = Value after reset

Table 11-12. SPI Flag Register (SPIFLG) Field Descriptions

Bit	Name	Value	Description
31–25	Reserved		Reads return zero and 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 BUF INIT ACTIVE 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		Reads return zero and writes have no effect.

Table 11-12. SPI Flag Register (SPIFLG) Field Descriptions (Continued)

Bit	Name	Value	Description
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.
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		Reads return zero and writes have no effect.

Table 11-12. SPI Flag Register (SPIFLG) Field Descriptions (Continued)

Bit	Name	Value	Description
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 SPIFLG register 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 (e.g. 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 SPIFLG register 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		Reads return zero and writes have no effect.

Table 11-12. SPI Flag Register (SPIFLG) Field Descriptions (Continued)

Bit	Name	Value	Description
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. • Set the SPIENA 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.
3	DESYNCFLG		Desynchronization of slave device. Desynchronization monitor is active in master mode only.
		0	No slave desynchronization 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. This flag can be cleared by one of the following methods. • Write a 1 to this bit. • Set SPIENA bit to 0.
2	PARITYERRFLG		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.
		0	No parity error detected.
		1	A parity error occurred. This flag can be cleared by one of the following methods. • Write a 1 to this bit. • Set SPIENA bit to 0.

Table 11-12. SPI Flag Register (SPIFLG) Field Descriptions (Continued)

Bit	Name	Value	Description
1	TIMEOUTFLG		Time-out caused by nonactivation of ENA signal.
		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, i.e. the SPI doesn't re-start a data transfer from this buffer. This flag can be cleared by one of the following methods. • Write a 1 to this bit. • Set SPIENA bit to 0.
0	DLEN ERR FLG		Data-length error flag. This flag can be cleared by one of the following methods. • Write a 1 to this bit. • Set SPIENA 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.

11.9.6 SPI Pin Control Register 0 (SPIPC0)

Note: Register bits vary by device
Register bits 31:24 and 23:16 of SPIPC0 to SPIPC8 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 11-6. SPI Pin Control Register 0 (SPIPC0) [offset = 14h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SOMIFUN[7:0]								SIMOFUN[7:-0]							
R/W-0								R/W-0							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				SOMIFUN0	SIMOFUN0	CLKFUN	ENAFUN	SCSFUN[7:0]							
R-0				R/W-0	R/W-0	R/W-0	R/W-UR/W-0	R/W-UR/W-0							

R = Read, W = write, -*n* = Value after reset

Table 11-13. SPI Pin Control (SPIPC0) Field Descriptions

Bit	Name	Value	Description
31-24	SOMIFUN[7:0]		Slave out, master in function. Determines whether SPISOMI[x] 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	SPISOMI[x] pin is a GIO pin.
		1	SPISOMI[x] pin is a SPI functional pin.
23-16	SIMOFUN[7:0]		Slave in, master out function. Determines whether SPISIMO[x] is to be used as a general-purpose I/O pin or as a SPI functional pin. **Note: Duplicate Control Bits for SPISIMO[x].** **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 SPISOMI[x] pin. The read value of Bit 16 always reflects the value of bit 10.**
		0	The SPISIMOx pin is a GIO pin.
		1	The SPISIMOx pin is a SPI functional pin
15–12	Reserved		Reads return zero and writes have no effect.

Table 11-13. SPI Pin Control (SPIPC0) Field Descriptions (Continued)

Bit	Name	Value	Description
11	SOMIFUN0		Slave out, master in function. This bit determines whether the SPISOMI0 pin is to be used as a general-purpose I/O pin or as a SPI functional pin.
		0	The SPISOMI0 pin is a GIO pin.
		1	The SPISOMI0 pin is a SPI functional pin.
			Note: Regardless of the number of parallel pins used, the SPISOMI0 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 SPISIMO0 pin is to be used as a general-purpose I/O pin or as a SPI functional pin.
		0	The SPISIMO0 pin is a GIO pin.
		1	The SPISIMO0 pin is a SPI functional pin.
			Note: Regardless of the number of parallel pins used, the SPISIMO0 pin will always have to be programmed as functional pins for any SPI transfers.
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	Reserved		Always write a 0 to this bit. Reads are undefined.
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	Reserved		Always write a 0 to these bits. Reads are undefined.
7-0	SCSFUN[7:0]		$\overline{\text{SPISCSx}}$ function. Determines whether each $\overline{\text{SPISCSx}}$ pin is to be used as a general-purpose I/O pin or as a SPI functional pin. If the slave $\overline{\text{SPISCSx}}$ pins are in functional mode and receive an inactive high signal, the slave SPI will place its output in high-z and disable shifting.
		0	The $\overline{\text{SPISCSx}}$ pin is a GIO pin.
		1	The $\overline{\text{SPISCSx}}$ pin is a SPI functional pin.

11.9.7 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 11-7. SPI Pin Control Register 1 (SPIPC1) [offset = 18h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SOMI DIR[7:0]								SIMO DIR[7:0]							
R/W-0								R/W-0							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				SOMI0 DIR	SIMO0 DIR	CLKDIR	ENADIR	SCSDIR[7:0]							
R-0				R/W-0	R/W-0	R/W-0	R/W-0U	R/W-0U							

R = Read, W = Write; -*n* = Value after reset

Table 11-14. SPI Pin Control Register (SPIPC1) Field Descriptions

Bit	Name	Value	Description
31–24	SOMIDIR[7:0]		SPISOMIx direction. Controls the direction of SPISOMIx when used for general-purpose I/O. If SPISOMIx 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 SPISOMI0.** **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 pin. The read value of Bit 24 always reflects the value of bit 11.**
		0	SPISOMIx pin is an input.
		1	SPISOMIx pin is an output.
23–16	SIMODIR[7:0]		SPISIMOx direction. Controls the direction of SPISIMOx when used for general-purpose I/O. If SPISIMOx 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 SPISIMO0.** **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 SPISOMI pin. The read value of Bit 16 always reflects the value of bit 10.**
		0	SPISIMOx pin is an input.
		1	SPISIMOx pin is an output.

Table 11-14. SPI Pin Control Register (SPIPC1) Field Descriptions (Continued)

Bit	Name	Value	Description
15–12	Reserved		Reads return zero and writes have no effect.
11	SOMIDIR0		SPISOMI0 direction. This bit controls the direction of the SPISOMI0 pin when it is used as a general-purpose I/O pin. If the SPISOMI0 pin is used as a SPI functional pin, the I/O direction is determined by the MASTER bit in the SPIGCR1 register.
		0	SPISOMI0 pin is an input.
		1	SPISOMI0 pin is an output.
10	SIMODIR0		SPISIMO0 direction. This bit controls the direction of the SPISIMO0 pin when it is used as a general-purpose I/O pin. If the SPISIMO0 pin is used as a SPI functional pin, the I/O direction is determined by the MASTER bit in the SPIGCR1 register.
		0	SPISIMO0 pin is an input.
		1	SPISIMO0 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	SPICLK pin is an input.
		1	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	$\overline{\text{SPIENA}}$ pin is an input.
		1	$\overline{\text{SPIENA}}$ pin is an output.
7–0	SCSDIR[7:0]		$\overline{\text{SPISCSx}}$ direction. These bits control the direction of each $\overline{\text{SPISCSx}}$ pin when it is used as a general-purpose I/O pin. Each pin could be configured independently from the others If the $\overline{\text{SPISCSx}}$ is used as a SPI functional pin; the I/O direction is determined by the CLKMOD bit (SPIGCR1[1]).
		0	$\overline{\text{SPISCSx}}$ pin is an input.
		1	$\overline{\text{SPISCSx}}$ pin is an output.

11.9.8 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 11-8. SPI Pin Control Register 2 (SPIPC2) [offset = 1Ch]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SOMIDIN[7:0]								SIMODIN[7:0]							
R-U								R-U							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				SOMIDIN0	SIMODIN0	CLKDIN	ENADIN	SCSDIN[7:0]							
R-0				R-U	R-U	R-U	R-U	R-U							

R = Read; W = Write; U = Undefined; -*n* = Value after reset

Table 11-15. SPI Pin Control Register 2 (SPIPC2) Field Descriptions

Bit	Name	Value	Description
31-24	SOMIDIN[7:0]		SPISOMIx data in. The value of the SPISOMIx pins.
		0	The SPISOMIx pin is logic 0.
		1	The SPISOMIx pin is logic 1.
23–16	SIMODIN[7:0]		SPISIMOx data in. The value of the SPISIMOx pins.
		0	The SPISIMOx pin is logic 0.
		1	The SPISIMOx pin is logic 1.
15–12	Reserved		Reads return zero and writes have no effect.
11	SOMIDIN0		SPISOMI0 data in. The value of the SPISOMI0 pin.
		0	The SPISOMI0 pin is logic 0.
		1	The SPISOMI0 pin is logic 1.
10	SIMODIN0		SPISIMO0 data in. The value of the SPISIMO0 pin.
		0	The SPISIMO0 pin is logic 0.
		1	The SPISIMO0 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

Table 11-15. SPI Pin Control Register 2 (SPIPC2) Field Descriptions (Continued)

Bit	Name	Value	Description
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[7:0]		$\overline{\text{SPISCSx}}$ data in. The value of the $\overline{\text{SPISCSx}}$ pins.
		0	The $\overline{\text{SPISCSx}}$ pin is logic 0.
		1	The $\overline{\text{SPISCSx}}$ pin is logic 1

11.9.9 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 11-9. SPI Pin Control Register 3 (SPIPC3) [offset = 20h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SOMIDOUT[7:0]								SOMIDOUT[7:0]							
R/W-U								R/W-U							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				SOMIDOUT0	SIMODOUT0	CLKD-OUT	ENADOUT	SCSDOUT[7:0]							
R-0				R/W-U	R/W-U	R/W-U	R/W-U	R/W-U							

R = Read; W = Write; -*n* = Value after reset

Table 11-16. SPI Pin Control Register 3 (SPIPC3) Field Descriptions

Bit	Name	Value	Description
31–24	SOMIDOUT[7:0]		SPISOMIx data out write. This bit is only active when the SPISOMIx 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 SPISOMI0. If a 32-bit write is performed, bit 11 will have priority over bit 24.**
		0	Current value on SPISOMIx pin is logic 0.
		1	Current value on SPISOMIx pin is logic 1
23–16	SIMODOUT[7:0]		SPISIMOx data out write. This bit is only active when the SPISIMOx 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 SPISOMI0. If a 32-bit write is performed, bit 10 will have priority over bit 16.**
		0	Current value on SPISIMOx pin is logic 0.
		1	Current value on SPISIMOx pin is logic 1.
15–12	Reserved		Reads return zero and writes have no effect.

Table 11-16. SPI Pin Control Register 3 (SPIPC3) Field Descriptions (Continued)

Bit	Name	Value	Description
11	SOMIDOUT0		SPISOMI0 data out write. This bit is only active when the SPISOMI0 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 SPISOMI0 pin is logic 0.
		1	Current value on SPISOMI0 pin is logic 1.
10	SIMODOUT0		SPISIMO0 data out write. This bit is only active when the SPISIMO0 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 SPISIMO0 pin is logic 0.
		1	The SPISIMO0 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.
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[7:0]		$\overline{\text{SPISCSx}}$ data out write. Only active when the $\overline{\text{SPISCSx}}$ pins are configured as a general-purpose I/O pins and configured as an output pins. The value of these bits indicates the value sent to the pins.
		0	The $\overline{\text{SPISCSx}}$ pin is logic 0.
		1	The $\overline{\text{SPISCSx}}$ pin is logic 1.

11.9.10 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 11-10. SPI Pin Control Register 4 (SPIPC4) [offset = 24h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SOMISET[7:0]								SOMISET[7:0]							
R/W-U								R/W-U							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				SOMISET0	SIMOSET0	CLKSET	ENASET	SCSSET[7:0]							
R-0				R/W-U	R/W-U	R/W-U	R/W-U	R/W-U							

R = Read; W = Write; -*n* = Value after reset

Table 11-17. SPI Pin Control Register 4 (SPIPC4) Field Descriptions

Bit	Name	Value	Description
31–24	SOMISET[7:0]		SPISOMIx data out set. This pin is only active when the SPISOMIx pin is configured as a general-purpose output pin. **Bit 11 or bit 24 can be used to set the SOMI0 pin. If a 32-bit write is performed, bit 11 will have priority over bit 24.**
		0	*Read:* SPISIMOx is logic 0. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* SPISOMIx is logic 1. *Write:* Logic 1 is placed on SPISOMIx pin if it is in general-purpose output mode.
23-17	Reserved		Reads return zero and writes have no effect.
23–16	SIMOSET[7:0]		SPISIMOx data out set. This bit is only active when the SPISIMOx pin is configured as a general-purpose output pin. **Bit 10 or bit 16 can be used to set the SOMI0 pin. If a 32-bit write is performed, bit 10 will have priority over bit 16.**
		0	*Read:* SPISIMIx is logic 0. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* SPISIMOx is logic 1. *Write:* Logic 1 is placed on SPISIMOx pin if it is in general-purpose output mode.
15–12	Reserved		Reads return zero and writes have no effect.

Table 11-17. SPI Pin Control Register 4 (SPIPC4) Field Descriptions (Continued)

Bit	Name	Value	Description
11	SOMISET0		SPISOMI0 data out set. This pin is only active when the SPISOMI0 pin is configured as a general-purpose output pin.
		0	*Read:* SPISOMI0 is logic 0. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* SPISOMI0 is logic 1. *Write:* Logic 1 is placed on SPISOMI0 pin if it is in general-purpose output mode.
10	SIMOSET0		SPISIMO0 data out set. This pin is only active when the SPISIMO0 pin is configured as a general-purpose output pin.
		0	*Read:* SPISIMO0 is logic 0. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* SPISIMO0 is logic 1. *Write:* Logic 1 is placed on SPISIMO0 pin if it is in general-purpose output mode.
9	CLKSET		SPICLK data out set. This bit is only active when the SPICLK pin is configured as a general-purpose output pin.
		0	*Read:* SPICLK is logic 0. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* SPICLK pin is logic 1. *Write:* Logic 1 is placed on the SPICLK pin if it is in general-purpose output mode.
8	ENASET		$\overline{\text{SPIENA}}$ data out set. This bit is only active when the $\overline{\text{SPIENA}}$ pin is configured as a general-purpose output pin.
		0	*Read:* SPIENA is logic 0. *Write:* A write to this bit has no effect.
		1	*Read:* SPIENA is logic 1. *Write:* Logic 1 is placed on $\overline{\text{SPIENA}}$ pin if it is in general-purpose O/P mode.
7–0	SCSSET[7:0]		$\overline{\text{SPISCSx}}$ data out set. This bit is only active when the $\overline{\text{SPISCSx}}$ pin is configured as a general-purpose output pin. A value of 1 written to this bit sets the corresponding $\overline{\text{SCS}}$DOUT bit to 1.
		0	*Read:* $\overline{\text{SPISCSx}}$ is logic 0. *Write:* A write to this bit has no effect.
		1	*Read:* $\overline{\text{SPISCSx}}$ is logic 1. *Write:* Logic 1 placed on $\overline{\text{SPISCSx}}$ pin if it is in general-purpose output mode.

11.9.11 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 11-11. SPI Pin Control Register 5 (SPIPC5) [offset = 28h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SOMICLR[7:0]								SIMOCLR[7:0]							
R/W-U								R/W-U							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				SOMICLR0	SIMOCLR0	CLKCLR	ENACLR	SCSCLR[7:0]							
R-0				R/W-U	R/W-U	R/W-U	R/W-0U	R/W-0U							

R = Read; W = Write; -*n* = Value after reset

Table 11-18. SPI Pin Control Register 5 (SPIPC5) Field Descriptions

Bit	Name	Value	Description
31–24	SOMICLR[7:0]		SPISOMIx data out clear. This pin is only active when the SPISOMIx pin is configured as a general-purpose output pin. **Bit 11 or bit 24 can be used to clear the pin SOMI0. If a 32-bit write is performed, bit 11 will have priority over bit 24.**
		0	*Read:* The current value on SOMIDOUTx is 0. Write: A write of 0 to this bit has no effect.
		1	*Read:* The current value on SOMIDOUTx is 1. *Write:* Logic 0 is placed on SPISOMIx pin if it is in general-purpose output mode.
23–16	SIMOCLR[7:0]		SPISIMOx data out clear. This bit is only active when the SPISIMOx pin is configured as a general-purpose output pin. **Bit 10 or bit 16 can be used to clear the pin SOMI0. If a 32-bit write is performed, bit 10 will have priority over bit 16.**
		0	*Read:* The current value on SIMODOUTx is 0. Write: A write of 0 to this bit has no effect.
		1	*Read:* The current value on SIMODOUTx is 1. *Write:* Logic 0 is placed on SPISIMOx pin if it is in general-purpose output mode.
15–12	Reserved		Reads return zero and writes have no effect.

Table 11-18. SPI Pin Control Register 5 (SPIPC5) Field Descriptions (Continued)

Bit	Name	Value	Description
11	SOMICLR0		SPISOMI0 data out clear. This bit is only active when the SPISOMI0 pin is configured as a general-purpose output pin.
		0	*Read:* The current value on SPISOMI0 is 0. Write: A write of 0 to this bit has no effect.
		1	*Read:* The current value on SPISOMI0 is 1. *Write:* Logic 0 is placed on SPISOMIx pin if it is in general-purpose output mode.
10	SIMOCLR0		SPISIMO0 data out clear. This bit is only active when the SPISIMO0 pin is configured as a general-purpose output pin.
		0	*Read:* The current value on SPISIMO0 is 0. Write: A write of 0 to this bit has no effect.
		1	*Read:* The current value on SPISIMO0 is 1. *Write:* Logic 0 is placed on SPISIMO0 pin if it is in general-purpose output mode.
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: A write of 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 ENA is 0. Write: A write of 0 to this bit has no effect.
		1	*Read:* The current value on $\overline{\text{ENA}}$ is 1. *Write:* Logic 0 is placed on $\overline{\text{SPIENA}}$ pin if it is in general-purpose output mode.
7–0	SCSCLR[7:0]		$\overline{\text{SPISCSx}}$ data out clear. This bit is only active when the $\overline{\text{SPISCSx}}$ pins are configured as a general-purpose output pins.
		0	*Read:* The current value on SCSDOUTx is 0. Write: A write of 0 to this bit has no effect.
		1	*Read:* The current value on $\overline{\text{SCSDOUT}}$x is 1. *Write:* Logic 0 is placed on the $\overline{\text{SPISCS}}$ pin if it is in general-purpose output mode.

11.9.12 SPI Pin Control Register 6 (SPIPC6)

Note: Register bits vary by device
Register bits 31:24 and 23:16 of SPIPC0 to SPIPC8 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 11-12. SPI Pin Control Register 6 (SPIPC6) [offset = 2Ch]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SOMIPDR[7:0]								SIMOPDR[7:0]							
R/W-0								R/W-0							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				SOMIP DR0	SIMOP DR0	CLK-PDR	ENAPDR	SCSPDR[7:0]							
R-0				R/W-0	R/W-0	R/W-0	R/W-0	R/W-0							

R = Read; W = Write; -*n* = Value after reset

Table 11-19. SPI Pin Control Register 6 (SPIPC6) Field Descriptions

Bit	Name	Value	Description
31–24	SOMIPDR[7:0]		SPISOMIx open drain enable. This bit enables open drain capability for the SPISOMIx pin if the following conditions are met: • SOMIDIRx = 1 (SPISOMIx pin configured in GIO mode as an output) • SOMIDOUTx = 1 **Bit 11 or bit 24 can both be used to enable open-drain for SOMI0. If a 32-bit write is performed, bit 11 will have priority over bit 24.**
		0	The output value on the SPISOMIx pin is logic 1.
		1	Output pin SPISOMIx is in a high-impedance state.
23-16	SIMOPDR[7:0]		SPISIMOx open drain enable. This bit enables open drain capability for the SPISIMOx pin if the following conditions are met: • SIMODIRx = 1 (SPISIMOx pin configured in GIO mode as an output) • SIMODOUTx = 1 **Bit 10 or bit 16 can both be used to enable open-drain for SIMO0. If a 32-bit write is performed, bit 10 will have priority over bit 16.**
		0	The output value on SPISIMOx pin is logic 1.
		1	Output pin SPISIMOx is in a high-impedance state.
15–12	Reserved		Reads return zero and writes have no effect.

Table 11-19. SPI Pin Control Register 6 (SPIPC6) Field Descriptions (Continued)

Bit	Name	Value	Description
11	SOMIPDR0		SOMI0 open-drain enable. This bit enables open-drain capability for SOMI0 if the following conditions are met. • SOMI0 pin configured in GIO mode as output pin • Output value on SPISOMI0 pin is logic 1.
		0	Output value 1 of SPISOMI0 pin is logic 1.
		1	Output value 1 of SPISOMI0 is high-impedance.
10	SIMOPDR0		SPISIMO0 open-drain enable. This bit enables open -drain capability for the SPISIMO0 pin if the following conditions are met. • SIMO0 pin configured in GIO mode as output pin • Output value on SPISIMO0 pin is logic 1.
		0	Output value 1 of SPISIMO0 pin is logic 1.
		1	Output value 1 of SPISIMO0 is high-impedance.
9	CLKPDR		CLK open drain enable. This bit enables open drain capability for the pin CLK if the following conditions are met: • SPICLK pin configured in GIO mode as an output pin • SPICLKDOUT = 1
		0	Output value on CLK pin is logic 1.
		1	Output pin CLK is in a high-impedance state.
8	ENAPDR		SPIENA pin open drain enable. This bit enables open drain capability for SPIENA if the following conditions are met: • SPIENA pin configured in GIO mode as an output pin • SPIENADOUT = 1
		0	Output value on SPIENA pin is logic 1.
		1	Output pin SPIENA is in a high-impedance state.
7–0	SCSPDR[7:0]		SPISCSx open drain enable. This bit enables open drain capability for the SPISCSx pin if the following conditions are met: • SPISCS pin configured in GIO mode as an output pin • SPISCSDOUTx = 1
		0	Output value on SCSx pin is logic 1.
		1	Output pin SCSx is in a high-impedance state.

11.9.13 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 11-13. SPI Pin Control Register 7 (SPIPC7) [offset = 30h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SOMIDIS[7:0]								SIMODIS[7:0]							
R/W-n								R/W-n							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				SOMIP DIS0	SIMOP DIS0	CLKDIS	ENAPDIS	SCSPDIS[7:0]							
R-0				R/W-n	R/W-n	R/W-n	R/W-nU	R/W-nU							

R = Read; W = Write; -*n* = Value after reset

Table 11-20. SPI Pin Control Register 7 (SPIPC7) Field Descriptions

Bit	Name	Value	Description
31–24	SOMIDIS[7:0]		SOMIx pull control enable/disable. This bit enables pull control capability for the SOMIx 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 SOMIO. If a 32-bit write is performed, bit 11 will have priority over bit 24.**
		0	Pull control on the SPISOMIx pin is enabled.
		1	Pull control on the SPISOMIx pin is disabled.
23–16	SIMODIS[7:0]		SIMOx pull control enable/disable. This bit enables pull control capability for the SIMOx 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 SIMO0. If a 32-bit write is performed, bit 10 will have priority over bit 16.**
		0	Pull control on SPISIMOx pin is enabled.
		1	Pull control on SPISIMOx pin is disabled.
15–12	Reserved		Reads return zero and writes have no effect.

Table 11-20. SPI Pin Control Register 7 (SPIPC7) Field Descriptions (Continued)

Bit	Name	Value	Description
11	SOMIPDIS0		SPISOMI0 pull control enable/disable. This bit enables pull control capability for the pin SPISOMI0 pin if it is in input mode regardless of whether it is in functional or GIO mode.
		0	Pull control on the SPISOMI0 pin is enabled.
		1	Pull control on the SPISOMI0 pin is disabled.
10	SIMOPDIS0		SPISIMO0 pull control enable/disable. This bit enables pull control capability for the pin SPISIMO0 pin if it is in input mode regardless of whether it is in functional or GIO mode.
		0	Pull control on SPISIMO0 pin is enabled.
		1	Pull control on SPISIMO0 pin is disabled.
9	CLKPDIS		CLK pull control enable/disable. This bit enables pull control capability for the pin SPICLK pin if it is in input mode regardless of whether it is in functional or GIO mode.
		0	Pull control on CLK pin is enabled.
		1	Pull control on CLK pin is disabled.
8	ENAPDIS		ENABLE pull control enable/disable. This bit enables pull control capability for the pin SPIENA pin if it is in input mode regardless of whether it is in functional or GIO mode.
		0	Pull control on ENABLE pin is enabled.
		1	Pull control on ENABLE pin is disabled.
7–0	SCSPDIS[7:0]		SCSx pull control enable/disable. This bit enables pull control capability for the pin SPISCSx pin if it is in input mode regardless of whether it is in functional or GIO mode.
		0	Pull control on SCSx pin is enabled.
		1	Pull control on SCSx pin is disabled.

11.9.14 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 11-14. SPI Pin Control Register 8 (SPIPC8) [offset = 34h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SOMIPSEL[7:0]								SIMOPSEL[7:0]							
R/W-n								R/W-n							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				SOMI-PSEL	SIMO-PSEL	CLKPSEL	ENAPSEL	SCSPSEL[7:0]							
R-0				R/W-n	R/W-n	R/W-n	R/W-	R/W-nU							

R = Read; W = Write; -*n* = Value after reset

Table 11-21. SPI Pin Control Register 8 (SPIPC8) Field Descriptions

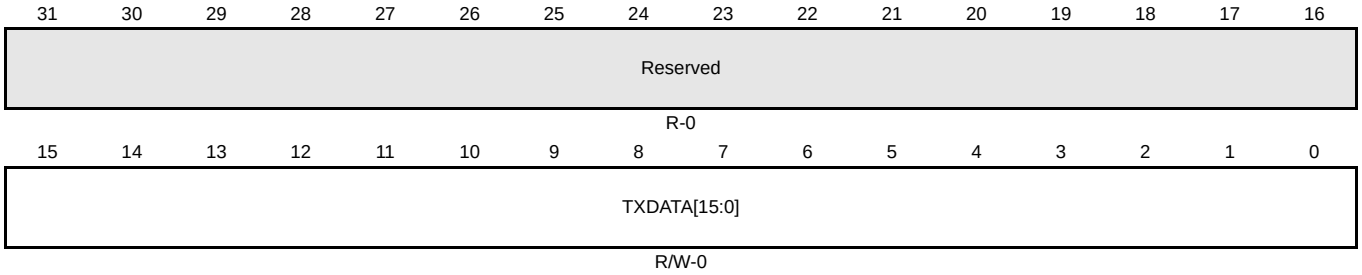
Bit	Name	Value	Description
31–24	SOMIPSEL[7:0]		SPISOMIx pull select. This bit selects the type of pull logic at the SOMIx pin. **Note: Bit 11 or bit 24 can be used to set pull-select for SPISOMI0. If a 32-bit write is performed, bit 11 will have priority over bit 24.**
		0	Pull down on SOMIx pin
		1	Pull up on SOMIx pin
23–16	SIMOPSEL[7:0]		SPISIMOx pull select. This bit selects the type of pull logic at the SPISIMOx pin. **Note: Bit 10 or bit 16 can be used to set pull-select for SPISOMI0. If a 32-bit write is performed, bit 10 will have priority over bit 16.**
		0	Pull down on SPISIMOx pin
		1	Pull up on SPISIMOx pin
15–12	Reserved		Reads return zero and writes have no effect.

Table 11-21. SPI Pin Control Register 8 (SPIPC8) Field Descriptions

Bit	Name	Value	Description
11	SOMIPSEL		SOMI pull select. This bit selects the type of pull logic at the SOMI pin.
		0	Pull down on the SPISOMI pin
		1	Pull up on the SOMI pin
10	SIMOPSEL		SPISIMO pull select. This bit selects the type of pull logic at the SPISIMO pin.
		0	Pull down on SPISIMO pin
		1	Pull up on SPISIMO pin
9	CLKPSEL		CLK pull select. This bit selects the type of pull logic at the CLK pin.
		0	Pull down on CLK pin
		1	Pull up on CLK pin
		0	Input buffer for CLK is disabled if PULLDIS = 1
		1	Input buffer for CLK is enabled if PULLDIS = 1
8	ENAPSEL		ENABLE pull select. This bit selects the type of pull logic at the ENABLE pin.
		0	Pull down on ENABLE pin.
		1	Pull up on ENABLE pin.
7–0	SCSPSEL[7:0]		SCSx pull select. This bit selects the type of pull logic at the SCSx pin.
		0	Pull down on SCSx.
		1	Pull up on SCSx pin.

11.9.15 SPI Transmit Data Register 0 (SPIDAT0)

Figure 11-15. SPI Transmit Data Register 0 (SPIDAT0) [offset = 38h]



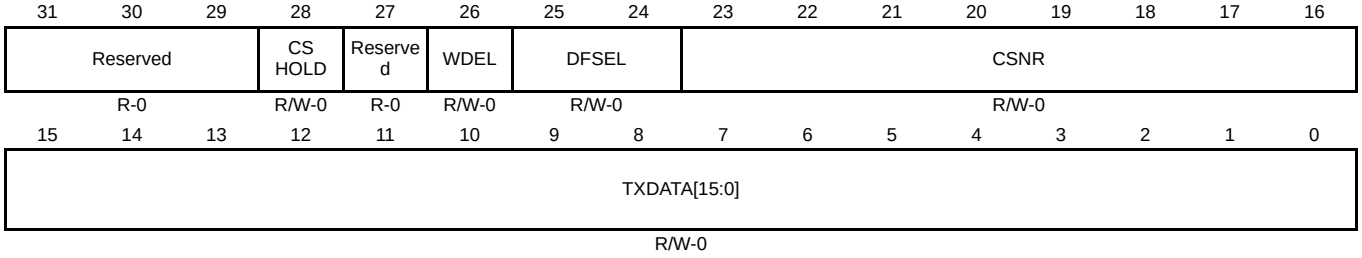
R = Read, W = write, -*n* = Value after reset

Table 11-22. SPI Transmit Data Register 0 (SPIDAT0) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Reads return zero and writes have no effect.
15–0	TXDATA(15–0)	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 register 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 SPIDAT1register.** **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.**

11.9.16 SPI Transmit Data Register 1 (SPIDAT1)

Figure 11-16. SPI Transmit Data Register 1 (SPIDAT1) [offset = 3Ch]



R = Read, W = write, -*n* = Value after reset

Table 11-23. SPI Transmit Data Register 1 (SPIDAT1) Field Descriptions

Bit	Name	Value	Description
31–29	Reserved		Reads return zero and 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		Reads return zero and 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, $\overline{\text{SPISCS}}$ pins will still be de-activated for at least for 2VCLK cycles if CSHOLD = 0. **Note: The duration for which the SPISCS 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 SPISCS 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 $\overline{\text{SPISCS}}$ pins will be de-activated for at least (WDELAY + 2) * VCLK_Period duration.

Table 11-23. SPI Transmit Data Register 1 (SPIDAT1) Field Descriptions (Continued)

Bit	Name	Value	Description
25–24	DFSEL		Data word format select
		00	Data word format 0 is selected
		01	Data word format 1 is selected
		10	Data word format 2 is selected
		11	Data word format 3 is selected
23–16	CSNR	0–FFh	Chip select number. CSNR defines the chip-select that will be activated during the data transfer. **Note: Writing to only the control field 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 fields in the control field to select the required phase/polarity combination.**
15–0	TXDATA(15–0)	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. Write to this register ONLY when using the automatic slave chip-select feature (see Section 11.2, *Operating Modes* on page 448 for more information). A write to this register will drive the contents of CSNR[3:0] on the SPISCS[3:0] pins, if they are configured as functional pins. 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.**

11.9.17 SPI Receive Buffer Register (SPIBUF)

Figure 11-17. SPI Receive Buffer Register (SPIBUF) [offset = 40h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXEMPTY	RX OVR	TX FULL	BIT ERR	DE SYNC	PARITYERR	TIME OUT	DLEN-ERR	LCSNR							
R-1	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDATA[15:0)]															
R-0															

R = Read, W = write, C = Clear; S = Set; -*n* = Value after reset

Table 11-24. SPI Receive Buffer Register (SPIBUF) Field Descriptions

Bit	Name	Value	Description
31	RXEMPTY		**Receive data buffer empty.** When the host reads the SPIBUF field or the entire SPIBUF register, it automatically sets the RXEMPTY flag. When a data transfer is completed, the received data is copied into SPIBUF, and the RXEMPTY flag is cleared.
		0	New data has been received and copied into the SPIBUF field.
		1	No data has been received since the last read of SPIBUF. This flag gets set to 1 under the following conditions: • Reading the RXDATA portion of the SPIBUF register. • Writing a 1 to clear the RXINTFLG bit in the SPIFLG register. 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 portion of SPIBUF (or the entire register).

Table 11-24. SPI Receive Buffer Register (SPIBUF) Field Descriptions (Continued)

Bit	Name	Value	Description
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 (e.g. CPU, DMA, or other host processor). If enabled, the RXOVRN interrupt is generated when RXBUF is overwritten, and reading either 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). This flag is cleared to 0 when the RXDATA 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 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.
28	BITERR		**Bit error.** There was a mismatch of internal transmit data and transmitted data.
		0	No bit error occurred. **Note: This flag is cleared to 0 when the RXDATA portion of the SPIBUF register is read.**

Table 11-24. SPI Receive Buffer Register (SPIBUF) Field Descriptions (Continued)

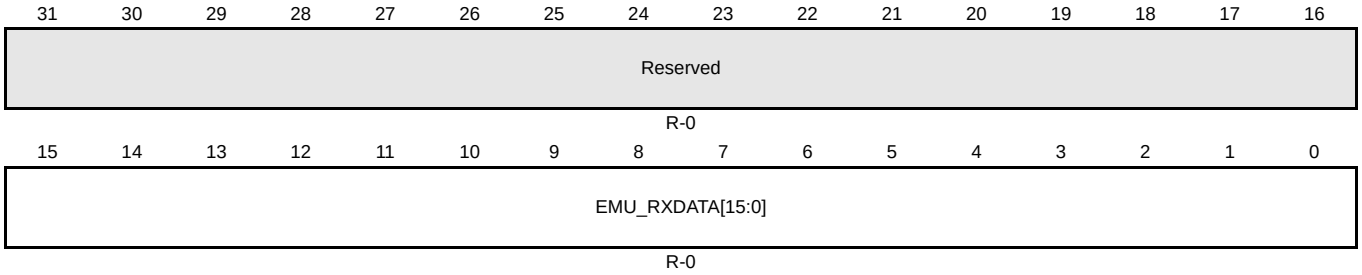
Bit	Name	Value	Description
		1	A bit error occurred. The SPI samples the signal of the transmit pins (master: SIMOx, slave: SOMIx) at the receive point (one-half clock cycle after the transmit point). If the sampled value differs from the transmitted value, a bit error is detected and the BITERR flag 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.**
		0	No slave desynchronization detected. **Note: This flag is cleared to 0 when the RXDATA portion of the SPIBUF register is read.**
		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 portion of the SPIBUF register is read.**
		0	No parity error detected.
		1	A parity error occurred.

Table 11-24. SPI Receive Buffer Register (SPIBUF) Field Descriptions (Continued)

Bit	Name	Value	Description
25	TIMEOUT		**Time-out because of non-activation of ENA 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 SPIFLG register is set. **This bit is valid only in master mode.** **This flag is cleared to 0 when RXDATA portion of the SPIBUF register is read.**
		0	No ENA-pin time-out occurred.
		1	An ENA signal time-out occurred.
24	DLENERR		**Data length error flag.** **Note: This flag is cleared to 0 when the RXDATA portion of the SPIBUF register is read.**
		0	No data-length error has occurred.
		1	A data length error has occurred.
23–16	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[15:0]	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.

11.9.18 SPI Emulation Register (SPIEMU)

Figure 11-18. SPI Emulation Register (SPIEMU) [offset = 44h]



R = Read, -*n* = Value after reset

Table 11-25. SPI Emulation Register (SPIEMU) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Reads return zero and writes have no effect.
15–0	EMU_RXDATA [15:0]	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.

11.9.19 SPI Delay Register (SPIDELAY)

Figure 11-19. SPI Delay Register (SPIDELAY) [offset = 48h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
C2TDELAY[7:0]								T2CDELAY[7:0]							
R/W-0								R/W-0							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T2EDELAY[7:0]								C2EDELAY[7:0]							
R/W-0								R/W-0							

R = Read, -*n* = Value after reset

Table 11-26. SPI Delay Register (SPIDELAY) Field Descriptions

Bit	Name	Value	Description
31–24	C2TDELAY[74:0]	0-1FF	**Chip-select-active to transmit-start delay.** See Figure 11-20 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 **Note: If C2TDELAY = 0, then $t_{C2TDELAY}$ = 0.** 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 11-26. SPI Delay Register (SPIDELAY) Field Descriptions (Continued)

Bit	Name	Value	Description
23–16	T2CDELAY[74:0]	0–1FF	**Transmit-end-to-chip-select-inactive-delay.** See Figure 11-21 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 **Note: If T2CDELAY = 0, then $t_{T2CDELAY}$ = 0** 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.** **Note: C2TDELAY and T2CDELAY counters do not count when SPIENA stops transmission.** Both C2TDELAY and T2CDELAY counters do not have any dependency on the SPIENA pin value. Even if the 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 SPIENA pin is deasserted by the slave, the master will continue to hold the SPISCS pins active until the T2CDELAY counter overflows. In this way, it is guaranteed that the setup and hold times of the SPISCS 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.

Table 11-26. SPI Delay Register (SPIDELAY) Field Descriptions (Continued)

Bit	Name	Value	Description
15–8	T2EDELAY[7:0]	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 SPIENA signal has to become inactive and after SPISCS 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 isn't disabled. The T2EDELAY defines a time-out value that triggers the DESYNC flag, if the SPIENA signal isn't deactivated in time. The DESYNC flag is set to indicate that the slave device did not de-assert its SPIENA pin in time to acknowledge that it received all bits of the sent word. See Figure 11-22 for an example of this condition. **Note: DESYNC is also set if the SPI detects a de-assertion of 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[7:0]	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 11-23 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 = 30 h; > $t_{C2EDELAY}$=6 ms; The slave device has to activate the ENA signal within 6 ms after the SPI has activated the chip select signal (SPISCS), otherwise the TIMEOUT flag is set and an interrupt is asserted (if enabled).

Figure 11-20. Example: $t_{C2TDELAY}$ = 8 VCLK Cycles

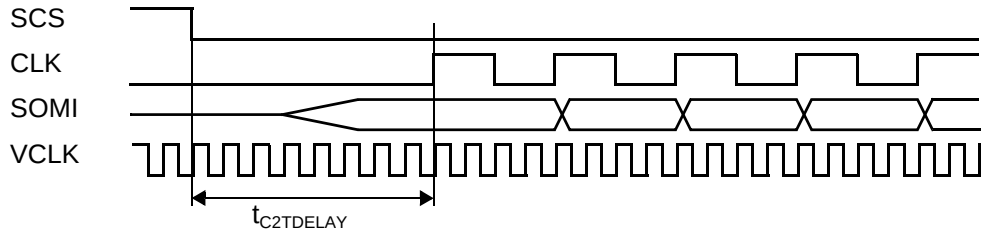


Figure 11-21. Example: $t_{T2CDELAY}$ = 4 VCLK Cycles

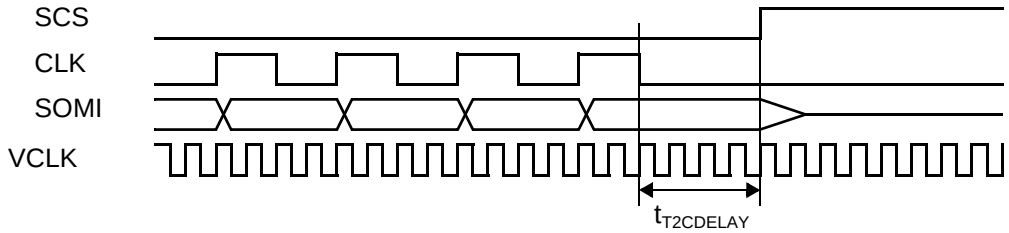


Figure 11-22. Transmit-Data-Finished-to-ENA-Inactive-Timeout

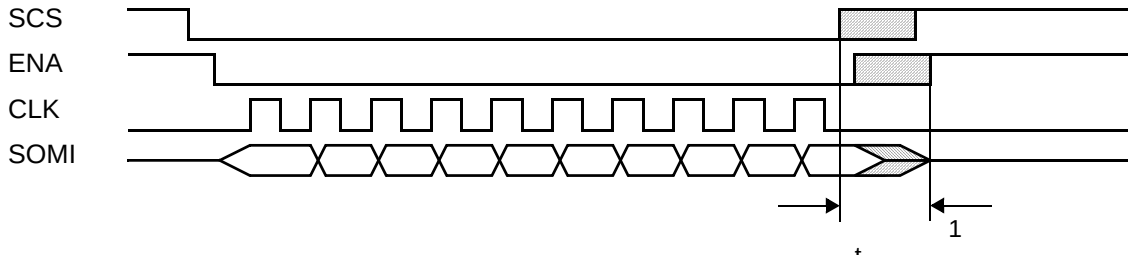
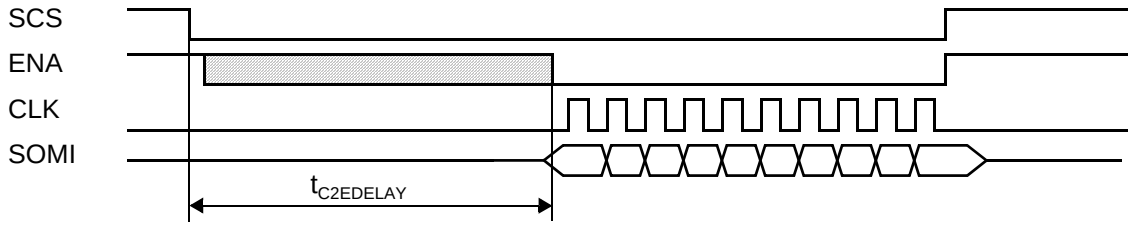


Figure 11-23. Chip-Select-Active-to-ENA-Signal-Active-Timeout



11.9.20 SPI Default Chip Select Register (SPIDEF)

Figure 11-24. SPI Default Chip Select Register (SPIDEF) [offset = 4Ch]

31–8	7–0
Reserved	
R-0	
Reserved	CSDEF
R-0	R/W-1

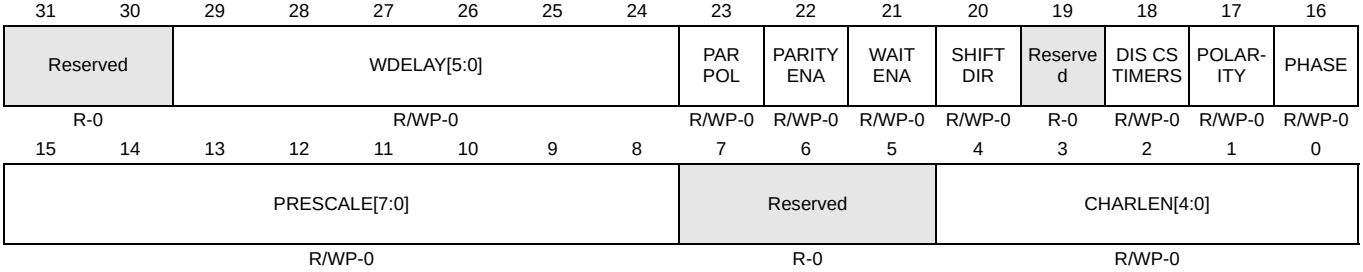
R = Read, W = Write; -*n* = Value after reset

Table 11-27. SPI Default Chip Select Register (SPIDEF) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return zero and writes have no effect.
7–0	CSDEF	0-FFh	**Chip select default pattern.** Master-mode only. The CSDEFx bits are output to the SPISCS 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	$\overline{\text{SPISCS}}$x is set to 0 when no transfer is active.
		1	$\overline{\text{SPISCS}}$x is set to 1 when no transfer is active.

11.9.21 SPI Data Format Registers (SPIFMT[3:0])

Figure 11-25. SPI Data Format Registers (SPIFMT[3:0]) [offset = 5Ch–50h]



R = Read, WP = Write in privilege mode only; -*n* = Value after reset

Table 11-28. SPI Data Format Registers (SPIFMT[3:0]) Field Descriptions

Bit	Name	Value	Description
31–30	Reserved		Reads return zero and writes have no effect.
29–24	WDELAY[5:0]	0–3F	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.
		0	No parity generation/ verification is performed for this data format.
		1	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. **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 SOMI 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.**

Table 11-28. SPI Data Format Registers (SPIFMT[3:0]) Field Descriptions (Continued)

Bit	Name	Value	Description
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.
19-18	Reserved		Reads return zero and writes have no effect.
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.

Table 11-28. SPI Data Format Registers (SPIFMT[3:0]) Field Descriptions (Continued)

Bit	Name	Value	Description
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: 1. In 3-pin/4-pin with nENA 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. 2. 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 SPI-CLK polarity may be incorrectly treated as a clock edge by some slaves.
		0	If POLARITYx is set to 0 the SPI clock signal is low-inactive, i.e., before and after data transfer the clock signal is low.
		1	If POLARITYx is set to 1 the SPI clock signal is high-inactive, i.e., 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 set 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[7:0]		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: $BR_{Formatx} = \frac{VBUSPCLK}{(PRESCALEx + 1)}$ **Note: When PRESCALEx is set to 0, the SPI clock rate defaults to VCLK/2.**
7–5	Reserved		Reads return zero and writes have no effect.
4–0	CHARLEN[4:0]	0–1F	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 0x10 (data word length = 16). Illegal values, such as 0x00 or 0x1F are not allowed; their effect is indeterminate.

11.9.22 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 11-26. *Interrupt Vector 0* (*INTVECT0*) [offset = 60h]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										INTVECT0					SUSP-END0
R-0										R-0					R-0

R = Read, WP = Write in privilege mode only; -*n* = Value after reset

Table 11-29. *Transfer Group Interrupt Vector 0* (*INTVECT0*)]

Bit	Name	Value	Description
31–6	Reserved		Reads return zero and 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.
		00000	There is no pending interrupt.
		00001 + x	Transfer group x (x=0,..,15) has a pending interrupt. SUSPEND0 reflects the type of interrupt (*suspended* or *finished*).
		10001	Error Interrupt pending. The lower half of SPIFLG contains more details about the type of error.
		10011	The pending interrupt is a “Receive Buffer Overrun” interrupt.
		10010	**SPI mode:** The pending interrupt is a “Receive Buffer Full” interrupt. **Mib mode:** Reserved. This bit combination should not occur.
		10100	**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.

Table 11-29. *Transfer Group Interrupt Vector 0* (*INTVECT0*)] (Continued)

Bit	Name	Value	Description
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 SPIFLG register. Writing a new word to the SPIDATx register clears the “Transmit Empty” interrupt.

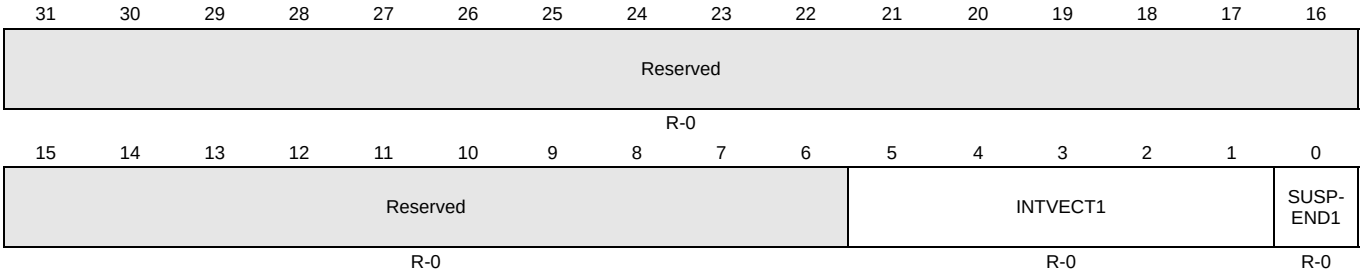
Note: In multi-buffer mode, INTVECT0 contains the interrupt for the highest priority transfer group. A read from INTVECT0 automatically causes the next-highest priority transfer group’s interrupt status to get loaded into INTVECT0 and its corresponding SUSPEND flag to get loaded into SUSPEND0. The transfer 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 SPIFLG register or by reading the RXOVERN_BUF_ADDR register.

11.9.23 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 11-27. *Interrupt Vector 1* (*INTVECT1*) [offset = 64h]



R = Read, WP = Write in privilege mode only; -*n* = Value after reset

Table 11-30. *Transfer Group Interrupt Vector 1* (*INTVECT1*)]

Bit	Name	Value	Description
31–6	Reserved		Reads return zero and 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.
		00000	There is no pending interrupt. **SPI mode only.**
		10001	Error Interrupt pending. The lower half of SPIINT1 contains more details about the type of error. **SPI mode only.**
		10011	The pending interrupt is a “Receive Buffer Overrun” interrupt. **SPI mode only.**
		10010	The pending interrupt is a “Receive Buffer Full” interrupt. **SPI mode only.**
		10100	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.**

Table 11-30. *Transfer Group Interrupt Vector 1* (*INTVECT1*)] (Continued)

Bit	Name	Value	Description
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 SPIFLG register. 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 SPIFLG register or by reading the RXOVERN_BUF_ADDR register.

11.9.24 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 (Bits31:24) are used when SPIFMT3 register is selected by DFSEL[1:0] = 11 in the control field of a buffer.

Figure 11-28. Parallel/Modulo Mode Control Register (SPIPMCTRL) [offset = 6Ch]

31–30	29	28–27–26	25–24	23–22	21	20–19–18	17–16
Reserved	MOD CLK POL3	MMODE 3(2–0[	PMODE 3(1–0)	Reserved	MOD CLK POL2	MMODE 2(2–0)	PMODE 2(1–0)
R-0	R/WP-0	R/WP-0	R/WP-0	R-0	R/WP-0	R/WP-0	R/WP-0

15–14	13	12–11–10	9–8	7–6	5	4–3–2	1–0
Reserved	MOD CLK POL1	MMODE 1(2–0)	PMODE 1(1–0)	Reserved	MOD CLK POL0	MMODE 0(2–0)	PMODE 0(1–0)
R-0	R/WP-0	R/WP-0	R/WP-0	R-0	R/WP-0	R/WP-0	R/WP-0

R = Read, W = write, P = Privilege mode, -*n* = Value after reset

Table 11-31. SPI Parallel/Modulo Mode Control Register (SPIPMCTRL) Field Descriptions

Bit	Name	Value	Description
31–30	Reserved		Reads return zeros and writes have no effect.
29	MOD CLK POL 3		Modulo mode SPICLK polarity. This bit determines the polarity of the SPICLK in modulo mode only. If the MODULO MODE[2:0] 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 11-31. SPI Parallel/Modulo Mode Control Register (SPIPMCTRL) Field Descriptions (Continued)

Bit	Name	Value	Description
28–26	MMODE 3(2–0)		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).
		000	Normal single dataline mode - Default (PMODE should be set to 00)
		001	2-data line mode (PMODE should be set to 00)
		010	3-data line mode (PMODE should be set to 00)
		011	4-data line mode (PMODE should be set to 00)
		100	5-data line mode (PMODE should be set to 00)
		101	6-data line mode (PMODE should be set to 01)
		110–111	Reserved
25–24	PMODE 3(1–0)		Parallel mode bits determine whether the SPI/MibSPI operates with 1, 2, 4 or 8 data lines.
		00	00 = normal operation/1-data line (MMODE should be set to 000)
		01	01 = 2-data line mode (MMODE should be set to 000)
		10	10 = 4-data line mode (MMODE should be set to 000)
		11	11 = 8-data line mode (MMODE should be set to 000)
23–22	Reserved		Reads return zeros and writes have no effect.
21	MOD CLK POL 2		Modulo mode SPICLK polarity. This bit determines the polarity of the SPICLK in modulo mode only. If the MMODE[2:0] 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	MMODE 2(2–0)		These bits determine whether the SPI/MibSPI operates with 1, 2, 4, 5, or 6 data lines (if the modulo option is supported by the module).
		000	000 = 1-data line Mode - Default (PMODE should be set to 00)
		001	001 = 2-data line Mode (PMODE should be set to 00)
		010	3-data line mode (PMODE should be set to 00)
		011	4-data line mode (PMODE should be set to 00)
		100	5-data line mode (PMODE should be set to 00)
		101	6-data line mode (PMODE should be set to 01)
		110–111	Reserved

Table 11-31. SPI Parallel/Modulo Mode Control Register (SPIPMCTRL) Field Descriptions (Continued)

Bit	Name	Value	Description
17–16	PMODE 2(1–0)		Parallel mode bits determine whether the SPI/MibSPI operates with 1, 2, 4 or 8 data lines.
		00	Normal operation/1-data line (MMODE should be set to 000)
		01	2-data line mode (MMODE should be set to 000)
		10	4-data line mode (MMODE should be set to 000)
		11	8-data line mode (MMODE should be set to 000)
15–12	Reserved		Reads return zeros and writes have no effect.
13	MOD CLK POL 1		Modulo mode SPICLK polarity. This bit determines the polarity of the SPICLK in modulo mode only. If the MMODE[2:0] 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	MMODE 1(2–0)		These bits determine whether the SPI/MibSPI operates with 1, 2, 4, 5, or 6 data lines (if the modulo option is supported by the module).
		000	1-data line mode - default (PMODE should be set to 00)
		001	2-data line mode (PMODE should be set to 00)
		010	3-data line mode (PMODE should be set to 00)
		011	4-data line mode (PMODE should be set to 00)
		100	5-data line mode (PMODE should be set to 00)
		101	6-data line mode (PMODE should be set to 01)
		110–111	Reserved
9–8	PMODE 1(1–0)		Parallel mode bits determine whether the SPI/MibSPI operates with 1, 2, 4 or 8 data lines.
		00	Normal operation/1-data line (MMODE should be set to 000)
		01	2-data line mode (MMODE should be set to 000)
		10	4-data line mode (MMODE should be set to 000)
		11	8-data line mode (MMODE should be set to 000)
7–6	Reserved		Reads return 0 and writes have no effect.

Table 11-31. SPI Parallel/Modulo Mode Control Register (SPIPMCTRL) Field Descriptions (Continued)

Bit	Name	Value	Description
5	MOD CLK POL 0		Modulo mode SPICLK polarity. Determines the polarity of the SPI-CLK in modulo mode only. If the MMODE[2:0] 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.
4–2	MMODE 0(2–0)		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).
		000	1-data line mode - default (PMODE should be set to 00)
		001	2-data line mode (PMODE should be set to 00)
		010	3-data line mode (PMODE should be set to 00)
		011	4-data line mode (PMODE should be set to 00)
		100	5-data line mode (PMODE should be set to 00)
		101	6-data line mode (PMODE should be set to 01)
		110–111	Reserved
1–0	PMODE 0(1–0)		Parallel mode bits determine whether the SPI/MibSPI operates with 1, 2, 4 or 8 data lines.
		00	Normal operation/1-data line (MMODE should be set to 000)
		01	2-data line mode (MMODE should be set to 000)
		10	4-data line mode (MMODE should be set to 000)
		11	8-data line mode (MMODE should be set to 000)

11.9.25 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 11-29. Multi-buffer Mode Enable Register (MIBSPIE) [offset = 70h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															RXRA-MAC-CESS
R-0															R/WP-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															MSPI-ENA
R-0															R/WP-0

R = Read, W = write, P = Privilege mode, *-n* = Value after reset

Table 11-32. Multi-buffer Mode Enable Register (MIBSPIE) Field Descriptions

Bit	Name	Value	Description
31–17	Reserved		Reads return zeros and writes have no effect.
16	RXRAMACCESS		**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 set 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		Reads return zeros and 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 MSPI-ENA is not set to 1, the multi-buffer mode registers are not writable.
		0	The SPI runs in compatibility mode, i.e., in this mode the MibSPI is fully code-compliant to the standard TMS470M Series 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.

11.9.26 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 11-30 and Table 11-33 represents a super-set device with the maximum number of TGs (16) assumed. The actual number of bits available varies per device.

Figure 11-30. TG Interrupt Enable Set Register (TGITENST) [offset = 74h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SETINTENRDY[15:0]															
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SETINTENSUS[15:0]															
R/W-0															

R = Read; W = Write; -*n* = Value after reset

Table 11-33. TG Interrupt Enable Set Register (TGITENST) Field Descriptions

Bit	Name	Value	Description
31-16	SET INTENRDY[15:0]		TG interrupt set (enable) when transfer finished.
		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	SET INTEN SUS[15:0]		TG interrupt set (enabled) when transfer suspended
		0	*Read:* The TGx-suspended 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-suspended interrupt is enabled. The interrupt gets generated when TGx is suspended. *Write:* Enable the TGx-suspended interrupt. The interrupt gets generated when TGx is suspended.

11.9.27 MibSPI 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 11-31 and Table 11-34 represents a super-set device with the maximum number of TGs (16) assumed. The actual number of bits available varies per device.

Figure 11-31. TG Interrupt Enable Clear Register (TGITENCR) [offset = 78h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CLR INTENRDY[15:0]															
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLR INTENSUS[15:0]															
R/W-0															

R = Read; W = Write; -*n* = Value after reset

Table 11-34. TG Interrupt Enable Clear Register (TGITENCR) Field Descriptions

Bit	Name	Value	Description
31-16	CLR INTENRDY[15:0]		TG interrupt clear (disabled) when transfer finished.
		0	*Read:* The TGx-completed interrupt is disabled. The 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.
15–0	CLR INTENSUS[15:0]		TG interrupt clear (disabled) when transfer suspended.
		0	*Read:* The TGx-suspended 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-suspended interrupt is enabled. The interrupt gets generated when TGx completes. *Write:* Disables the TGx-suspended interrupt.

11.9.28 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 11-32 and Table 11-35 represents a super-set device with the maximum number of TGs (16) assumed. The actual number of bits available varies per device.

Figure 11-32. Transfer Group Interrupt Level Set Register (TGITLVST) [offset = 7Ch]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SET INTLVLRDY[15:0]															
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SET INTLVLSUS[15:0]															
R/W-0															

R = Read; W = Write; -*n* = Value after reset

Table 11-35. Transfer Group Interrupt Level Set Register (TGITLVST) Field Descriptions

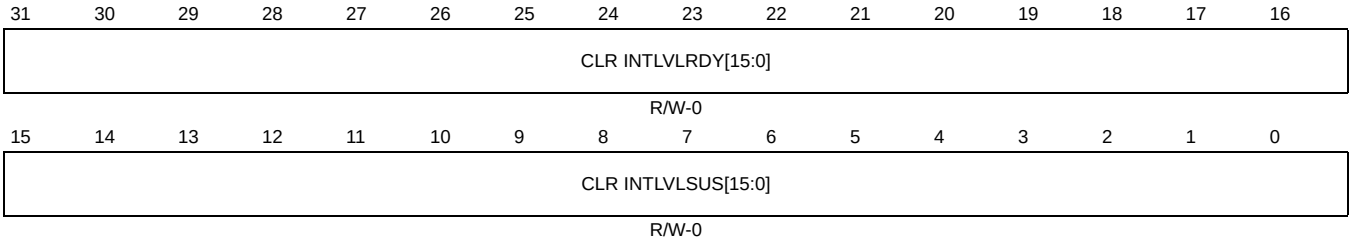
Bit	Name	Value	Description
31-16	SET INTLVLRDY[15:0]		Transfer-group completed interrupt level set.
		0	*Read:* The TGx-completed interrupt is set to INT0. *Write:* A write of 0 to this bit has no effect.
		1	*Read:* The TGx-completed interrupt is set to INT1. *Write:* Set the TGx-completed interrupt to INT1.
15–0	SET INTLVLSUS[15:0]		Transfer-group suspended interrupt level set.
		0	*Read:* 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 TG-x suspended interrupt INT1.

11.9.29 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 11-33 and Table 11-36 represents a super-set device with the maximum number of TGs (16) assumed. The actual number of bits available varies per device.

Figure 11-33. Transfer Group Interrupt Level Clear Register (TGITLVCR) [offset = 80h]



R = Read; W = Write; -*n* = Value after reset

Table 11-36. Transfer Group Interrupt Level Clear Register (TGITLVCR) Field Descriptions

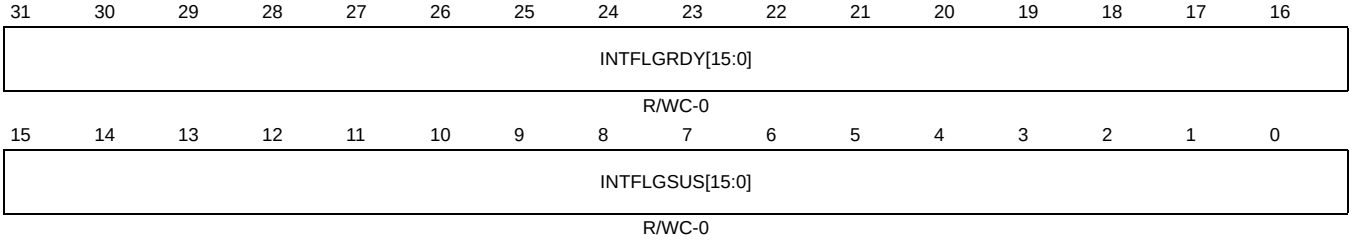
Bit	Name	Value	Description
31-16	CLR INTLVLRDY[15:0]		Transfer-group completed interrupt level clear.
		0	*Read:* The TGx-completed interrupt is set to INT0. *Write:* A write of 0 to this bit has no effect.
		1	*Read:* The TGx-completed interrupt is set to INT1. *Write:* Sets the TG-x completed interrupt to INT0.
15–0	CLR INTLVLSUS[15:0]		Transfer group suspended interrupt level clear.
		0	*Read:* The TG-x 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:* Sets the TGx-suspended interrupt to INT0.

11.9.30 Transfer Group Interrupt Flag Register (TGINTFLAG)

The TGINTFLAG 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 TGINTFLAG belongs to one TG.

The register map shown in Figure 11-34 and Table 11-37 represents a super-set device with the maximum number of TGs (16) assumed. The actual number of bits available varies per device.

Figure 11-34. Transfer Group Interrupt Flag Register (TGINTFLAG) [offset = 84h]



R = Read; W = Write; C = Clear; *-n* = Value after reset

Table 11-37. Transfer Group Interrupt Level Clear Register (TGITLVCR) Field Descriptions

Bit	Name	Value	Description
31-16	INTFLGRDY[15:0]		Transfer-group interrupt flag for a transfer-completed interrupt. **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.

Table 11-37. Transfer Group Interrupt Level Clear Register (TGITLVCR) Field Descriptions (Continued)

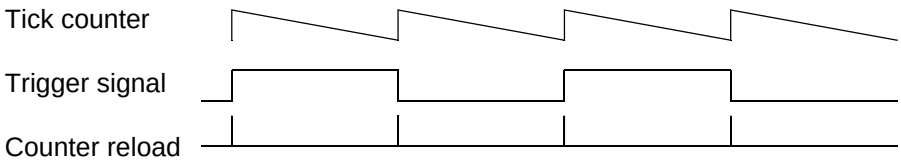
Bit	Name	Value	Description
15–0	INTFLGSUS[15:0]		Transfer-group interrupt flag for a transfer-suspend interrupt. **Note: Read Clear Behavior. Reading the interrupt vector registers TGINTVECT0 or TGINTVECT1 automatically clears the interrupt flag bit INTFLGSUSx referenced by the vector number given by 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.

11.9.31 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 11-35 as a square wave, illustrates the different trigger event types for the TGs (e.g., rising edge, falling edge, and both edges).

Figure 11-35. Tick Counter Operation



This register is shown in Figure 11-36 and described in Table 11-38.

Figure 11-36. Tick Count Register (TICKCNT) [offset = 90h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TICK ENA	RE LOAD	CLKCTRL(1–0)		Reserved											
R/W-0	R0/S-0	R/W-0		R-0											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TICKVALUE(15–0)															
R/W-0															

R = Read; W = Write; S = Set; C = Clear; *-n* = Value after reset;

Table 11-38. Tick Count Register (TICKCNT) Field Descriptions

Bit	Name	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[1:0]. When the tick counter is enabled it starts down-counting from its current value. When TICKENA goes from 0 to 1, the tick counter is automatically loaded with the contents of TICKVALUE.

Table 11-38. Tick Count Register (TICKCNT) Field Descriptions (Continued)

Bit	Name	Value	Description
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(1–0)		**Tick counter clock source control.** CLKCTRL defines the clock source that is used to clock the internal tick counter.
		00	SPICLK of data word format 0 is selected as the clock source of the tick counter
		01	SPICLK of data word format 1 is selected as the clock source of the tick counter
		10	SPICLK of data word format 2 is selected as the clock source of the tick counter
		11	SPICLK of data word format 3 is selected as the clock source of the tick counter
27–16	Reserved		Reads return 0 and writes have no effect.
15–0	TICKVALUE(15–0)	0–FFFF	**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.

11.9.32 Last TG End Pointer (LTGPEND)

Figure 11-37. Last TG End Pointer (LTGPEND) [offset = 94h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			TG IN SERVICE(4–0)					Reserved							
R-0			R-0					R-0							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	LPEND(6–0)							Reserved							
R-0	R/W-0							R-0							

R = Read, W = Write, C = Clear, -*n* = Value after reset, x = indeterminate

Table 11-39. Last TG End Pointer (LTGPEND) Field Descriptions

Bit	Name	Value	Description
31–29	Reserved		Reads return 0 and writes have no effect.
28–24	TG IN SERVICE(4–0)		**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.
		00000	No TG is being serviced by the sequencer.
		00001	TG0 is being serviced by the sequencer.
		. . .	...
		10000	TG15 is being serviced by the sequencer. **Note: The number of transfer groups varies by device.**
		10001–11111	Invalid values
23–15	Reserved		Reads return 0 and writes have no effect.
14–8	LPEND(6–0)	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, i.e., no end address is defined. Therefore LPEND has to be programmed to specify explicitly the end address of the last TG. **Note: LPEND allows SW compatibility for MibSPI variants Because of software compatibility reasons, the last TG end address has to be configurable; otherwise a super-set MibSPI cannot emulate a MibSPI with less TGs without software changes.** **Note: The number of transfer groups varies by device, thus the last TG also varies by device.**
7–0	Reserved		Reads return 0 and writes have no effect.

11.9.33 TGx Control Registers (TGxCTRL)

Each TG can be configured via one dedicated control register. The register description below shows one control register(x) which is identical for all TGs. For example, the control register for TG 2 is named TG2CTRL and is located at *base address+98h+4*2*. The actual number of available control registers varies by device.

Figure 11-38. MibSPI TG Control Registers (TGxCTRL) [offsets = 98h–D4h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TG ENA[0:15]	ONE SHOT[0:15]	PRST[0:15]	TGTD[0:15]	Reserved				TRGEVT[0:15](3–0)				TRIGSRC[0:15](3–0)			
R/W-0	R/W-0	R/W-0	R-0	R-0				R/W-0				R/W-0			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	PSTART[0:15](6–0)							Reserved	PCURRENT[0:15](6–0)						
R-0	R/W-0							R-0	R-0						

R = Read; W = Write; -*n* = Value after reset;

Table 11-40. TG Control Registers (TGxCTRL) Field Descriptions

Bit	Name	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.

Table 11-40. TG Control Registers (TGxCTRL) Field Descriptions (Continued)

Bit	Name	Value	Description
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.
27–24	Reserved		Reads return 0 and writes have no effect.

Table 11-40. TG Control Registers (TGxCTRL) Field Descriptions (Continued)

Bit	Name	Value		Description
23–20	TRIGEVTx(3-0)			**Type of trigger event.** A level-triggered TG can be stopped by de-activating the level trigger. However, the following restrictions apply. 1. 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. 2. 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. 3. 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.
		0000	never	Never trigger TGx. This is the default value after reset.
		0001	rising edge	A rising edge (0 to 1) at the selected trigger source (TRIGSRCx) initiates a transfer for TGx
		0010	falling edge	A falling edge (1 to 0) at the selected trigger source (TRIGSRCx) initiates a transfer for TGx
		0011	both edges	Rising and falling edges at the selected trigger source (TRIGSRCx) initiates a transfer for TGx
		0100	Reserved	
		0101	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 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.**
		0110	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 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.**

Table 11-40. TG Control Registers (TGxCTRL) Field Descriptions (Continued)

Bit	Name	Value	Description	
		0111	always	A repetitive group transfer will be performed. **Note: By setting the TRIGSRC to 0000b, the TRIGEVT to 0111b (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.**
		1xxx	Reserved	

Table 11-40. TG Control Registers (TGxCTRL) Field Descriptions (Continued)

Bit	Name	Value	Description	
19–16	TRIGSRCx(3-0)		**Trigger source.** After reset, the trigger sources of all TGs are disabled.	
		0000	disabled	
		0001	EXT0	External trigger source 0. The actual source varies per device (e.g. HET I/O channel, event pin, etc.).
		0010	EXT1	External trigger source 1. The actual source varies per device (e.g. HET I/O channel, event pin, etc.).
		0011	EXT2	External trigger source 2. The actual source varies per device (e.g. HET I/O channel, event pin, etc.).
		0100	EXT3	External trigger source 3. The actual source varies per device (e.g. HET I/O channel, event pin, etc.).
		0101	EXT4	External trigger source 4. The actual source varies per device (e.g. HET I/O channel, event pin, etc.).
		0110	EXT5	External trigger source 5. The actual source varies per device (e.g. HET I/O channel, event pin, etc.).
		0111	EXT6	External trigger source 6. The actual source varies per device (e.g. HET I/O channel, event pin, etc.).
		1000	EXT7	External trigger source 7. The actual source varies per device (e.g. HET I/O channel, event pin, etc.).
		1001	EXT8	External trigger source 8. The actual source varies per device (e.g. HET I/O channel, event pin, etc.).
		1010	EXT9	External trigger source 9. The actual source varies per device (e.g. HET I/O channel, event pin, etc.).
		1011	EXT10	External trigger source 10. The actual source varies per device (e.g. HET I/O channel, event pin, etc.).
		1100	EXT11	External trigger source 11. The actual source varies per device (e.g. HET I/O channel, event pin, etc.).
		1101	EXT12	External trigger source 12. The actual source varies per device (e.g. HET I/O channel, event pin, etc.).
		1110	EXT13	External trigger source 13. The actual source varies per device (e.g. HET I/O channel, event pin, etc.).
		1111	TICK	Internal periodic event trigger. The tick counter can initiate periodic group transfers.
15	Reserved		Reads return 0 and writes have no effect.	

Table 11-40. TG Control Registers (TGxCTRL) Field Descriptions (Continued)

Bit	Name	Value	Description
14–8	PSTARTx(6-0)	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's start address minus one (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		Reads return 0 and writes have no effect.
6–0	PCURRENTx(6-0)	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, PCURRENTx contains the address of the currently suspended word. After the TG resumes from suspend to wait mode, the next buffer will be transferred; i.e. 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 which 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.

11.9.34 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, e.g., 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 11-39. DMA Channel Control Register (DMAxCTRL) [offset = D8–114h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ONE SHOTx	BUFIDx							RXDMA_MAPx				TXDMA_MAPx			
R/W-0	R/W-0							R/W-0				R/W-0			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RX DMA ENAx	TX DMA ENAx	NO BRKx	ICOUNTx					Reserved	COUNT BIT17x	COUNTx					
R/W-0	R/W-0	R/W-0	R/W-0					x	R-0	R-0					

R = Read, W = Write, C = Clear, -*n* = Value after reset, x = indeterminate

Table 11-41. DMA Channel Control Register (DMAxCTRL) Field Descriptions

Bit	Name	Value	Description
31	ONESHOT		Auto-disable of DMA channel after ICOUNT+1 transfers.
		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[3:0] 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[3:0] shall differ from TXDMA_MAPx[3:0] and shall differ from any other used physical DMA Request line. Otherwise unexpected interference may occur.

Table 11-41. DMA Channel Control Register (DMAxCTRL) Field Descriptions (Continued)

Bit	Name	Value	Description
19–16	TXDMA_MAPx	0–Fh	Each MibSPI DMA channel can be linked to two physical DMA Request lines of the DMA controller. ne request line for receive data and the other for request line for transmit data. TXDMA_MAPx[3:0] 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 then TXDMA_MAPx[3:0] shall differ from RXDMA_MAPx[3:0] 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.
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.

Table 11-41. DMA Channel Control Register (DMAxCTRL) Field Descriptions (Continued)

Bit	Name	Value	Description
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 to 000** **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[4:0]. Every time COUNTx hits zero it is reloaded with ICOUNTx. The real number of transfers equals ICOUNTx plus one. 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. **Note: See** Section 11.9.35, *DMAxCOUNT Register (ICOUNT)* **on page 559 and** Section 11.9.36, *DMA Large Count (DMACNTLEN)* **on page 560 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 chipselect between the buffers (if CSHOLD is used).**

Table 11-41. DMA Channel Control Register (DMAxCTRL) Field Descriptions (Continued)

Bit	Name	Value	Description
7	Reserved		Reads return zeros and 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.**

11.9.35 DMAxCOUNT Register (ICOUNT)

Note:
These registers are used only if the LARGE COUNT bit in the DMACNTLEN register is set.

Note:
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 11-40. DMAxCOUNT Register (ICOUNT) [offset = F8–114h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ICOUNTx[15:0]															
R/W-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
COUNTx[15:0]															
R-0															

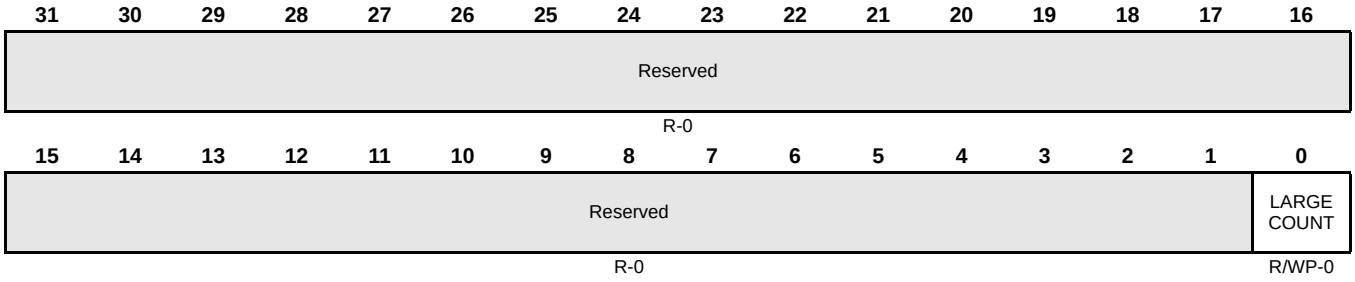
R = Read; W = Write; -*n* = Value after reset

Table 11-42. MibSPI DMAxCOUNT Register (ICOUNT) Field Descriptions

Bit	Name	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 zero, it is reloaded with ICOUNTx. The real number of transfer equals ICOUNTx plus one. 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	**The 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.**

11.9.36 DMA Large Count (DMACNTLEN)

Figure 11-41. DMA Large Count Register (DMACNTLEN) [offset = 118h]



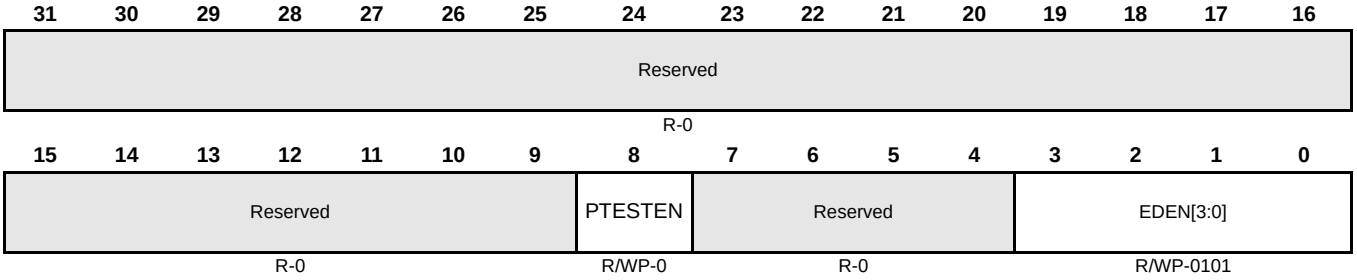
R = Read, -*n* = Value after reset

Table 11-43. MibSPI DMA Large Count Register (DMACNTLEN) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved		Reads return zero and 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.

11.9.37 Multi-buffer RAM *Uncorrectable Parity Error Control Register (UERRCTRL)*

Figure 11-42. Multi-buffer RAM Uncorrectable Parity Error Control Register (UERRCTRL) [offset = 120h]



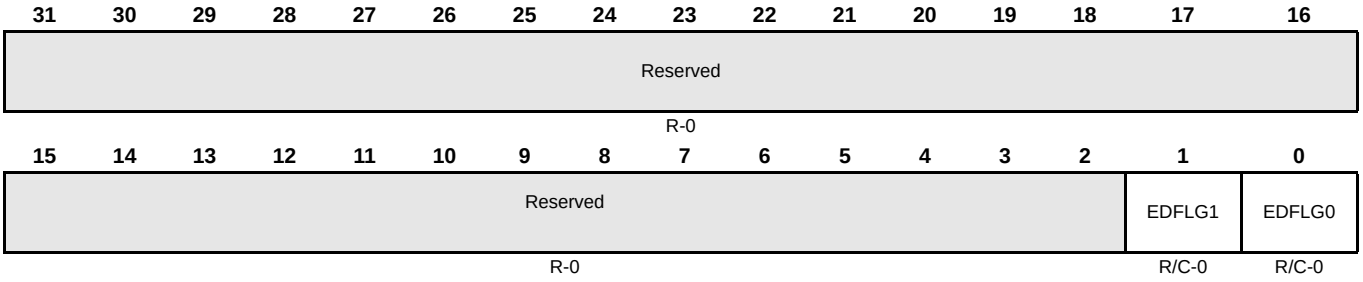
R = Read, WP = Write in Privilege mode only, -*n* = Value after reset

Table 11-44. Multi-buffer RAM Uncorrectable Parity Error Control Register (UERRCTRL) Field Descriptions

Bit	Name	Value	Description
31–9	Reserved		Reads return zero and 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 11.11, *Parity Memory* on page 579 for further details about parity memory testing.
		0	Parity bits are not memory-mapped.
		1	Parity bits are memory-mapped.
7–4	Reserved		Reads return zero and writes have no effect.
3–0	EDEN[3:0]		Error detection enable. These bits enable parity error detection.
		0101	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 detection.

11.9.38 Multi-buffer RAM Uncorrectable Parity Error Status Register (UERRSTAT)

Figure 11-43. Multi-buffer RAM Uncorrectable Parity Error Status Register (UERRSTAT) [offset = 124h]



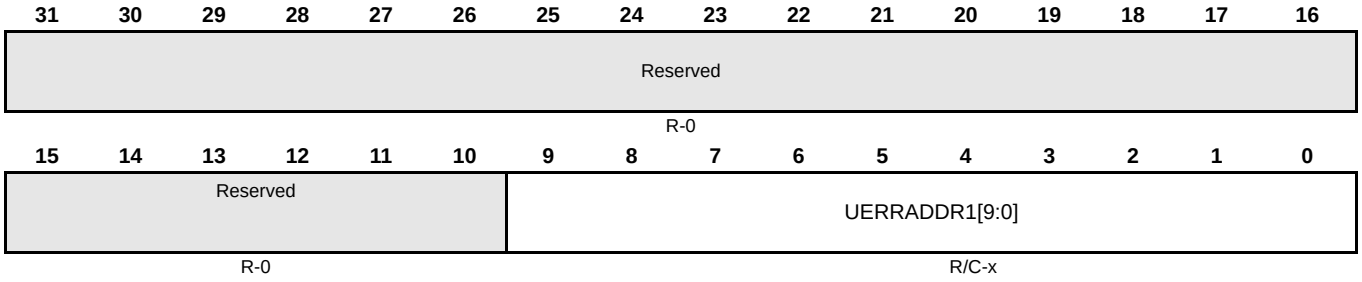
R = Read; C = Clear; -*n* = Value after reset

Table 11-45. Multi-buffer RAM Uncorrectable Parity Error Status Register (UERRSTAT) Field Descriptions

Bit	Name	Value	Description
31–2	Reserved		Reads return zero and 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 zero 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 zero to this bit has no effect.
		1	*Read:* An error was detected and the address is captured in the UERRADDR0 register. *Write:* The bit was cleared.

11.9.39 RXRAM Uncorrectable Parity Error Address Register (UERRADDR1)

Figure 11-44. RXRAM Uncorrectable Parity Error Address Register (UERRADDR1) [offset = 128h]



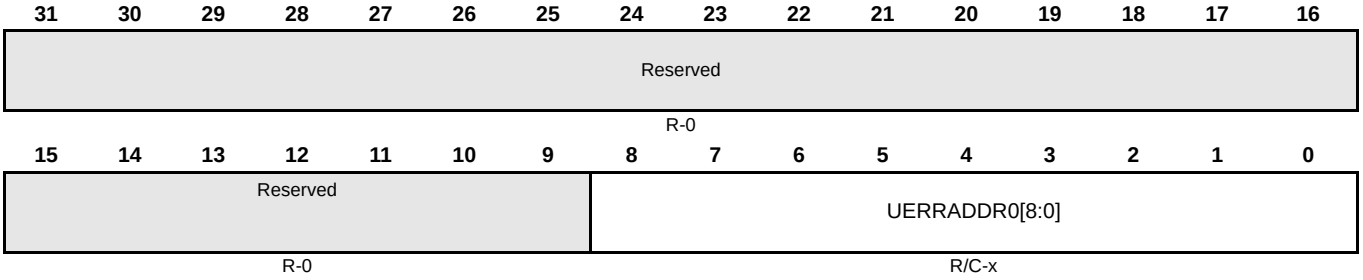
R = Read, C = Clear; -*n* = Value after power-on reset; -*x* = Interdeterminate

Table 11-46. RXRAM Uncorrectable Parity Error Address Register (UERRADDR1) Field Descriptions

Bit	Name	Value	Description
31–10	Reserved		Reads return zero and writes have no effect.
9–0	OVERRADDR1 [9:0]	200–3FFh	**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 0x200 - 0x3FF. The register does not clear its contents during or after module-level reset, system-level reset or even power-up reset. A read operation to this register clears its contents to the default value 0x200. After a power-up 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.**

11.9.40 TXRAM Uncorrectable Parity Error Address Register (UERRADDR0)

Figure 11-45. TXRAM Uncorrectable Parity Error Address Register (UERRADDR0) [offset = 12Ch]



R = Read; C-Clear; -*n* = Value after power-on reset; -*x* = Indeterminate

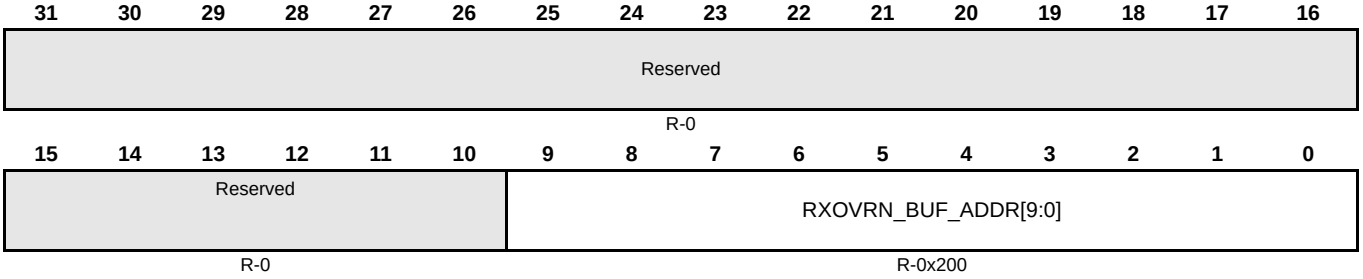
Table 11-47. TXRAM Uncorrectable Parity Error Address Register (UERRADDR0) Field Descriptions

Bit	Name	Value	Description
31–10	Reserved		Reads return zero and writes have no effect.
8–0	UERRADDR1 [8:0]	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 0x000 - 0x1FF. The register does not clear its contents during or after module-level reset, system-level reset, or even power-up reset. A read operation to this register clears its contents to all 0s. After a power-up 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.**

11.9.41 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 11-46. RXRAM Overrun Buffer Address Register (RXOVRN_BUF_ADDR) [offset = 130h]



R = Read, -*n* = Value after power-on reset

Table 11-48. RXRAM Overrun Buffer Address Register (RXOVRN_BUF_ADDR) Field Descriptions

Bit	Name	Value	Description
31–10	Reserved		Reads return zero and writes have no effect.
9–0	RXOVRN_BUF_ADDR[9:0]	200–3FCh	**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 0x200 - 0x3FC. Contents of this register are valid only when any of the TGINTVECT0 or TGINTVECT1 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.**

11.9.42 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 IO LPBK TST ENA field is set to 1010.

Figure 11-47. I/O-Loopback Test Control Register (IOLPBKTSTCR) [offset = 134h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved							SCS FAIL FLG	Reserved			CTRL BIT ERR	CTRLD ES YNC	CTRL PAR ERR	CTRL TIME OUT	CTRL DLEN ERR
R-0							R/C-0	R-0			R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				IOLPBKTSTENA[3:0]				Reserved		ERR SCS PIN			CTRL SCS PIN ERR	LPBK TYPE	RXP ENA
R-0				R/WP-0101				R-0		R/WP-000			R/WP-0	R/WP-0	R/WP-0

R = Read, WP = Write in Privilege mode only, C = Clear; -*n* = Value after power-on reset

Table 11-49. I/O-Loopback Test Control Register (IOLPBKTSTCR) Field Descriptions

Bit	Name	Value	Description
31–25	Reserved		Reads return zero and writes have no effect.
24	SCS FAIL FLAG		Bit indicating a failure on $\overline{\text{SPISCS}}$ 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 zero 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 $\overline{\text{SPISCS}}$[3:0] pins failed. A stuck-at fault is detected on one of the $\overline{\text{SPISCS}}$[3:0]. 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		Reads return zero and 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	Introduces bit errors by inverting the value of the incoming data during loopback.

Table 11-49. I/O-Loopback Test Control Register (IOLPBKTSTCR) Field Descriptions (Continued)

Bit	Name	Value	Description
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 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.
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 $\overline{\text{SPISCS}}$ pin (if functional) is forced to 1 when the module starts shifting data.
15–12	Reserved		Reads return zero and writes have no effect.
11–8	IOLPBKTSTENA[3:0]		Module I/O loopback test enable key.
		1010	Enable I/O loopback test mode.
		All other values	Disable I/O loopback test mode.
7–6	Reserved		Reads return zero and writes have no effect.

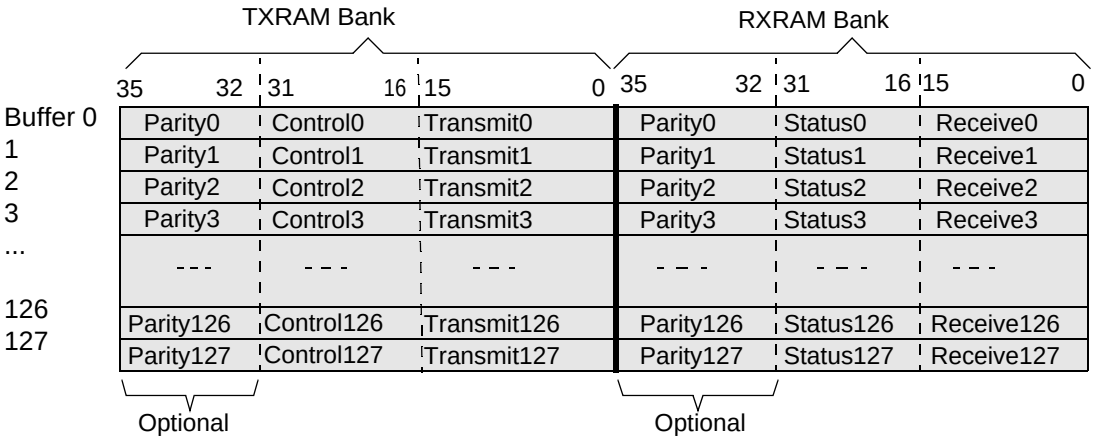
Table 11-49. I/O-Loopback Test Control Register (IOLPBKTSTCR) Field Descriptions (Continued)

Bit	Name	Value	Description
5–3	ERR SCS PIN[2:0]		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 chipselect pin selected by this field is forced to the opposite of its value in the CSNR.
		000	Select $\overline{\text{SPISCS}}$[0] for injecting error
		001	Select $\overline{\text{SPISCS}}$[1] for injecting error
		. . .	. . .
		111	Select $\overline{\text{SPISCS}}$[7] for injecting error.
2	CTRL SCS PIN ERR		Enable/disable the injection of an error on the $\overline{\text{SPISCS}}$[3:0] pins. The individual $\overline{\text{SPISCS}}$[3:0] pins can be chosen using the ERR SCS PIN field.
		0	Disable the $\overline{\text{SPISCS}}$[3:0] error-inducing logic.
		1	Enable the $\overline{\text{SPISCS}}$[3:0] error-inducing logic.
1	LPBK TYPE		**Module I/O loopback type (analog/digital).** See Figure 11-23 for the different types of loopback modes.
		0	Enable Digital loopback when IOLPBKTSTENA = 1010.
		1	Enable Analog loopback when IOLPBKTSTENA = 1010.
0	RXP ENA		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.

11.10 Multi-Buffer RAM

The multi-buffer RAM is used for holding transit & 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 11-48. A 4-bit parity field per word is also included in each bank of RAM.

Figure 11-48. Multi-Buffer RAM Configuration



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 bit field CSNR[7:0] 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.

11.10.1 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’ (Multibuffer RAM initialization is complete) before configuring multi-buffer RAM.

Besides the default auto initialisation after reset, the auto-initialization sequence can also be done in following ways:

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 MSIENA 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 User Guide for more details on the memory auto-initialization process.

Note:
The multibuffer related registers might be cleared during the auto initialisation. Therefore, do NOT trigger auto Initialization after configuring any Multibuffer mode registers.

11.10.2 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 offset of the multi-buffer RAM is 0xFF0E 0000.

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
Transmit Buffers																
0x00h–1FCh Buffer[0:127]	BUFMODE(2–0)			CS HOLD	LOCK	WDEL	DFSEL(1–0)		CSNR(7–0)							
	TXDATA(15–0)															
Receive Buffers:																
0x200–3FCh Buffer[0:127]	RXEMPTY	RX OVR	TX FULL	BIT ERR	DE SYNC	PARITYERR	TIME OUT	DLEN ERR	LCSNR(7–0)							
	RXDATA(15–0)															

11.10.3 Multi-buffer RAM Transmit Data Register

Each word of TXRAM is a transmit-buffer register.

Figure 11-49. Multi-buffer RAM Transmit Data Register [offset = Base + 000–1FFh]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BUFMODE			CS HOLD	LOCK	WDEL	DFSEL[1:0]		CSNR[7:0]							
R/W-0			R/W-0	R/W-0	R/W-0	R/W-0		R/W-0							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXDATA[15:0]															
R/W-0															

R = Read, W = write, -*n* = Value after reset

Table 11-50. Multi-buffer RAM Transmit Data Register Field Descriptions

Bit	Name	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.
		000	**disabled.** The buffer is disabled
		001	**skip single-transfer mode.** Skip this buffer until the corresponding TXFULL flag is set (i.e. new transmit data is available).
		010	**skip overwrite-protect mode.** Skip this buffer until the corresponding RXEMPTY flag is set (i.e. new receive data can be stored in RXDATA without data loss).
		011	**skip single-transfer overwrite-protect mode**. Skip this buffer until both of the corresponding TXFULL and RXEMPTY flags are set. (i.e. new transmit data available and previous data received by the host).
		100	**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.
		101	**suspend single-transfer mode.** Suspend-to-wait until the corresponding TXFULL flag is set (i.e. the sequencer stops at the current buffer until new transmit data is written in the TXDATA field).

Table 11-50. Multi-buffer RAM Transmit Data Register Field Descriptions (Continued)

Bit	Name	Value	Description
		110	**suspend overwrite-protect mode.** Suspend-to-wait until the corresponding RXEMPTY flag is set (i.e. the sequencer stops at the current buffer until the previously-received data is read by the host.
		111	**suspend single-transfer overwrite-protect mode.** Suspend-to-wait until the corresponding TXFULL and RXEMPTY flags are set (i.e. 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 in compatibility-mode of MibSPI, (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	LOCK		Lock two consecutive buffer words. Do not allow interruption by TG's 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 will be ignored.**
		0	No delay will be inserted. However, $\overline{\text{SPISCS}}$ pins will still be de-activated for at least for 2VCLK cycles if CSHOLD = 0. **Note: The duration for which the SPISCS 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 SPISCS 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 $\overline{\text{SPISCS}}$ pins will be de-activated for at least (WDELAY + 2) * VCLK_Period duration.

Table 11-50. Multi-buffer RAM Transmit Data Register Field Descriptions (Continued)

Bit	Name	Value	Description
25–24	DFSEL[2:0]		Data word format select
		00	Data word format 0 is selected
		01	Data word format 1 is selected
		10	Data word format 2 is selected
		11	Data word format 3 is selected
23–16	CSNR[7:0]	0–FF	Chip select number. CSNR defines the chip-select that will be activated during the data transfer. **Note:** **1) Writing to only the control field (using byte writes) 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 fields in the control field to select the required phase/polarity combination.** **2) Bits [23:20] is not writable in the device due to non availability of Chip select pins CS[4:7].**
15–0	TXDATA[15:0]	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 SPIDAT1 to 0x0000. Write to this register ONLY when using the automatic slave chip-select feature (see Section 11.2, *Operating Modes* on page 448 for more information). A write to this register will drive the contents of CSNR[7:0] on the SPISCS[3:0] pins, if they are configured as functional pins. 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.**

11.10.4 Multi-buffer RAM Receive Buffer Register

Each word of RXRAM is a receive-buffer register.

Figure 11-50. Multi-buffer RAM Receive Buffer Register [offset = RAM Base + 200–3FFh]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXEMPTY	RX OVR	TX FULL	BIT ERR	DE SYNC	PARITYERR	TIME OUT	DLEN-ERR	LCSNR[7:0]							
RS-1	RC-0	R-0	RC-0	RC-0	RC-0	RC-0	RC-0	R-0							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDATA[15:0)]															
R/W-0															

R = Read, W = write, C = Clear; S = Set; -*n* = Value after reset

Table 11-51. Multi-buffer Receive Buffer Register Field Descriptions

Bit	Name	Value	Description
31	RXEMPTY		**Receive data buffer empty.** When the host reads the SPIBUF field or the entire SPIBUF register, it automatically sets the RXEMPTY flag. When a data transfer is completed, the received data is copied into SPIBUF, and the RXEMPTY flag is cleared.
		0	New data has been received and copied into the SPIBUF field.
		1	No data has been received since the last read of SPIBUF.
			This flag gets set to 1 under the following conditions: • Reading the RXDATA portion of the SPIBUF register. • Writing a 1 to clear the RXINTFLG bit in the SPIFLG register. 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 portion of SPIBUF (or the entire register).

Table 11-51. Multi-buffer Receive Buffer Register Field Descriptions

Bit	Name	Value	Description
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 (e.g. CPU, DMA, or other host processor). If enabled, the RXOVRN interrupt is generated when RXBUF is overwritten, and reading either 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). This flag is cleared to 0 when the RXDATA 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 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.
		11	The transmit buffer is full; SPIDAT0/SPIDAT1 is not ready to accept new data.

Table 11-51. Multi-buffer Receive Buffer Register Field Descriptions

Bit	Name	Value	Description
28	BITERR		**Bit error.** There was a mismatch of internal transmit data and transmitted data.
		0	No bit error occurred. **Note: This flag is cleared to 0 when the RXDATA portion of the SPIBUF register is read.**
		1	A bit error occurred. The MibSPI 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: There is a possible inconsistency of DESYNC in the 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 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.** **This inconsistency in the desync flag is valid only in the compatibility mode of MibSPI. In multi-buffer mode, the desync flag is always guaranteed to be for the current buffer.**
		0	No slave desynchronization detected. **Note: This flag is cleared to 0 when the RXDATA portion of the SPIBUF register is read.**
		1	A slave device is desynchronized.

Table 11-51. Multi-buffer Receive Buffer Register Field Descriptions

Bit	Name	Value	Description
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 portion of the SPIBUF register is read.**
		0	No parity error detected.
		1	A parity error occurred.
25	TIMEOUT		**Time-out because of non-activation of ENA pin.** The MibSPI 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 SPIFLG register is set. **This bit is valid only in master mode.** **This flag is cleared to 0 when RXDATA portion of the SPIBUF register is read.**
		0	No ENA-pin time-out occurred.
		1	An ENA signal time-out occurred.
24	DLENERR		**Data length error flag.** **Note: This flag is cleared to 0 when the RXDATA portion of the SPIBUF register is read.**
		0	No data-length error has occurred.
		1	A data length error has occurred.
23–16	LCSNR[7:0]	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. The updated after transmission during the write-back of received data.
15–0	RXDATA[15:0]	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.

11.11 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 11-51 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.

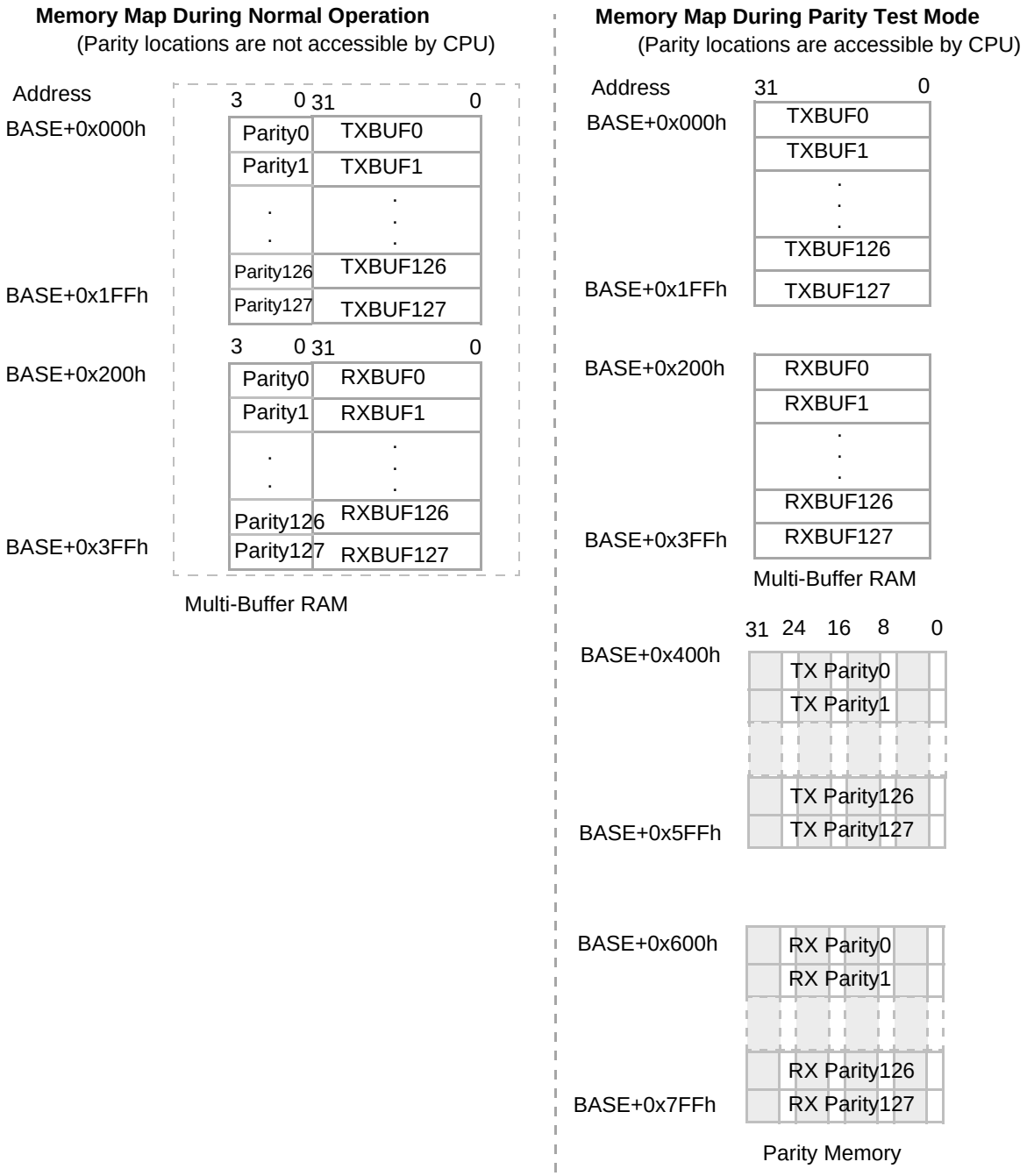
See the device-specific data sheet to get the actual base address of the multi-buffer RAM.

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 11-52 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 11-51. Memory Map for Parity Locations during Normal and Test Mode



* BASE - Base Address of Multi-Buffer RAM
Refer to specific Device Data sheet for the actual value of BASE.

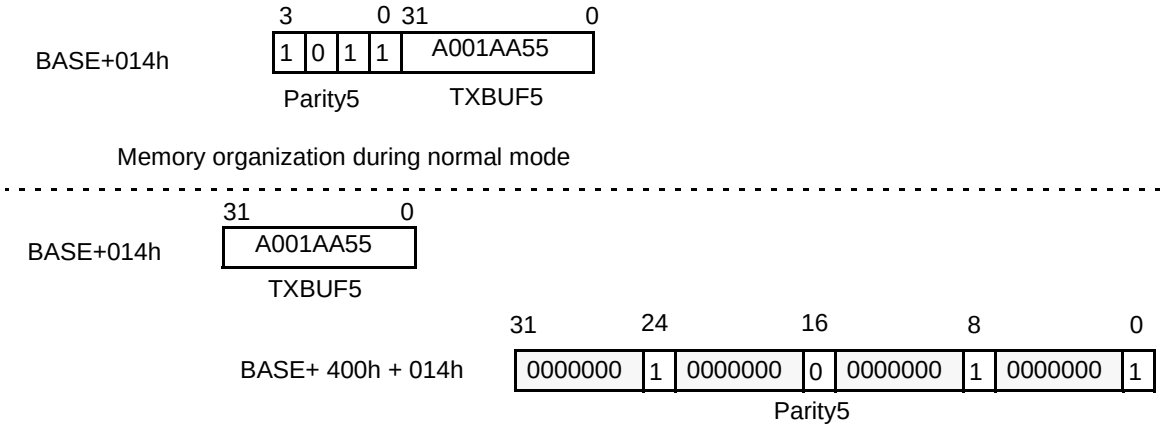
* Shaded areas indicate they are physically not present.

11.11.1 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 11-52 for a diagram.

Figure 11-52. Example of Memory Mapped Parity Locations during Test Mode



Parity memory locations during test mode (memory mapped)

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.

11.12 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 & 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
- $\overline{\text{SPISCS}}$[7:0] (SPI Slave Chip Selects) - Output
- SPISOMI (SPI Slave Out Master In) - Input
- $\overline{\text{SPIENA}}$ (SPI slave ready Enable) - Input

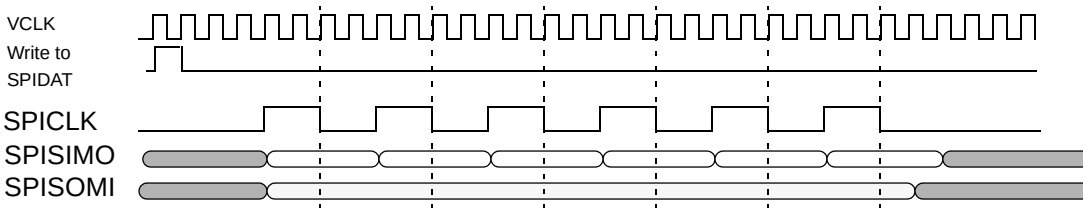
Slave mode SPI:

- SPICLK - Input
- SPISIMO - Input
- $\overline{\text{SPISCS}}$ - Input
- SPISOMI - Output
- $\overline{\text{SPIENA}}$ - Output

All the timing diagrams given below are with Phase='0' & Polarity = '0' unless explicitly stated otherwise.

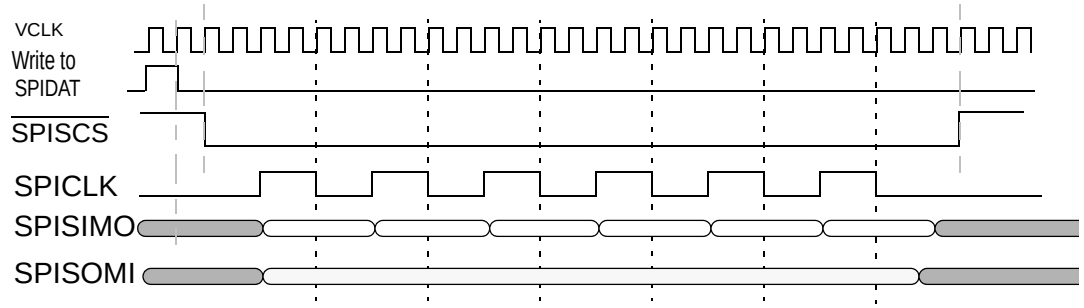
11.12.1 Master Mode Timings for SPI/MibSPI

Figure 11-53. SPI/MibSPI Pins during Master mode 3-pin configuration



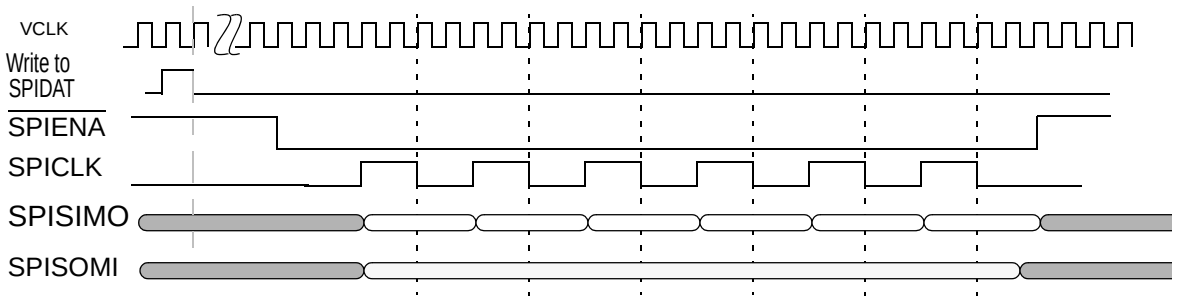
* Dotted vertical lines indicate the receive edges

Figure 11-54. SPI/MibSPI pins during Master mode 4-pin with $\overline{\text{SPISCS}}$ configuation.



* Dotted vertical lines indicate the receive edges

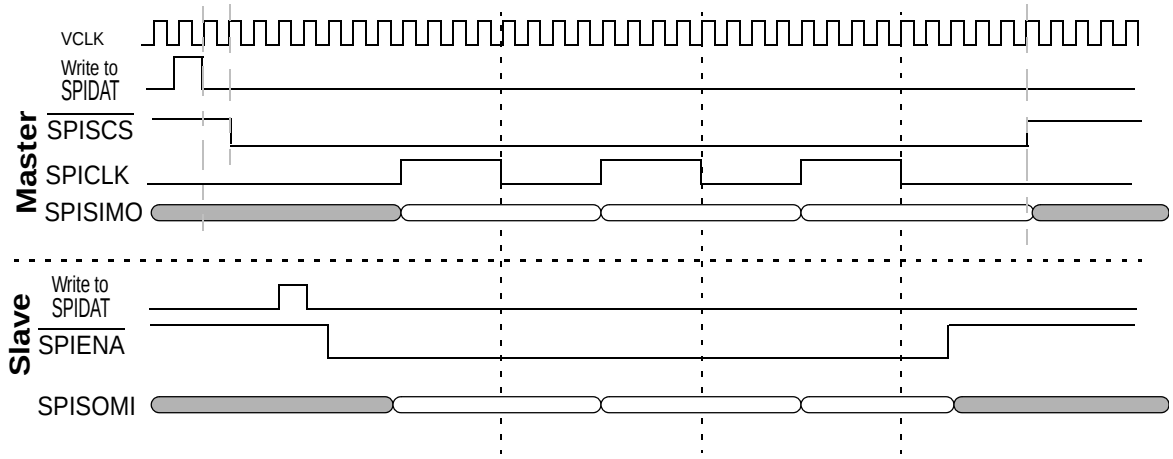
Figure 11-55. SPI/MibSPI pins during Master mode in 4pin with $\overline{\text{SPIENA}}$ configuration



* De-activation of $\overline{\text{SPIENA}}$ pin is controlled by the Slave.

* Dotted vertical lines indicate the receive edges

Figure 11-56. SPI/MibSPI pins during Master/Slave mode with 5-pin configuration

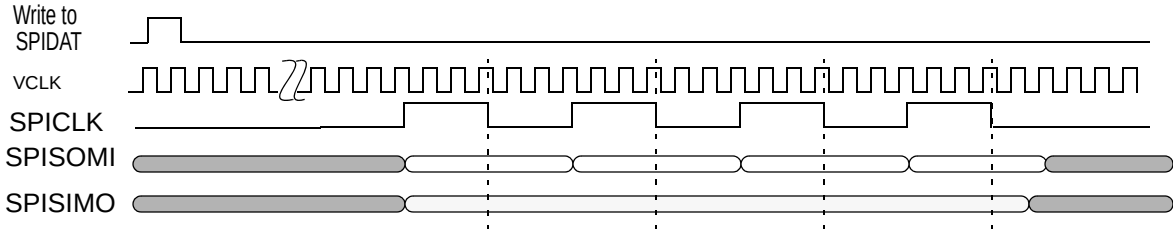


* Dotted vertical lines indicate the receive edges for the Master

* ENABLE_HIGHZ is set to '0' in Slave SPI

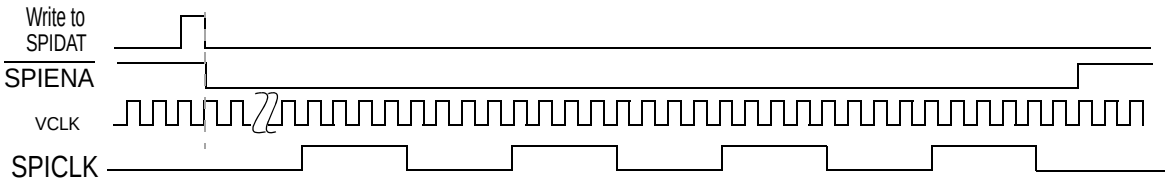
11.12.2 Slave Mode Timings for SPI/MibSPI

Figure 11-57. SPI/MibSPI pins during Slave mode 3-pin configuration



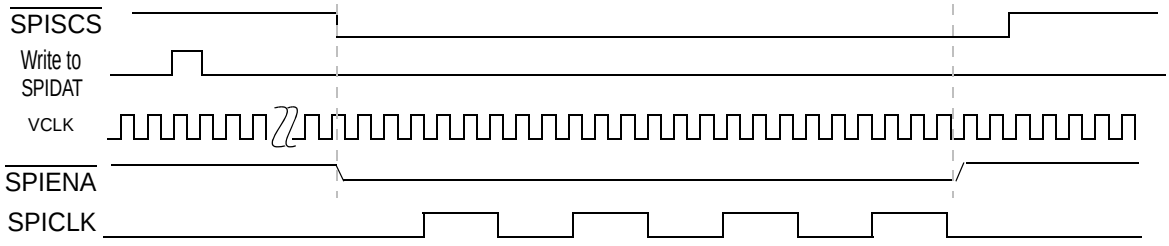
* Dotted vertical lines indicate the receive edges

Figure 11-58. SPI/MibSPI pins during Slave mode in 4pin with $\overline{\text{SPIENA}}$ configuration



* Diagram shows a relationship between the $\overline{\text{SPIENA}}$ from Slave and SPICLK from Master

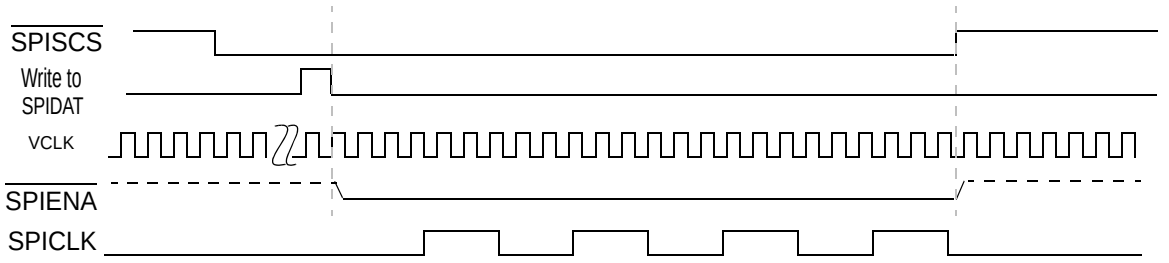
Figure 11-59. SPI/MibSPI pins during Slave mode in 5pin configuration - (Single Slave).



* ENABLE_HIGHZ is set to '0' in Slave SPI

* Diagram shows relationship between the $\overline{\text{SPISCS}}$ from a Master to $\overline{\text{SPIENA}}$ from Slave SPI when $\overline{\text{SPIENA}}$ is configured in Push-Pull mode

Figure 11-60. SPI/MibSPI pins during Slave mode in 5pin configuration - (Single/Multi Slave).



* ENABLE_HIGHZ is set to '1' in Slave SPI

* Diagram shows relationship between the $\overline{\text{SPISCS}}$ from a Master to $\overline{\text{SPIENA}}$ from Slave SPI when $\overline{\text{SPIENA}}$ is configured in High-Z mode

11.12.3 Timing Parameters of SPI/MibSPI pins in all the modes.

Table 11-52. Timing parameters of SPI/MibSPI pins.

SI No.	Delay Item	Master	Slave	Note
1	SPICLK edge to SPISIMO	0	-	
2	SPICLK edge to SPISOMI	-	0	
3a	$\overline{\text{SPISCS}}$ - SPICLK edge Phase = '0' (4pin configuration- Master, CSHOLD = '0')	2VCLK	-	Delay between $\overline{\text{SPISCS}}$ assertion by the Master to the first edge of its SPICLK. This timing is based on CSHOLD being '0' for the current and the previous buffer tranfer
3b	$\overline{\text{SPISCS}}$ - SPICLK edge Phase = '0' (4pin configuration- Master, CSHOLD = '1')	3VCLK	-	Delay between $\overline{\text{SPISCS}}$ assertion by the Master to the first edge of its SPICLK. This timing is based on CSHOLD being '1' for the current and the previous buffer tranfer
4a	$\overline{\text{SPISCS}}$ - SPICLK edge Phase = '1' (4pin configuration- Master, CSHOLD = '0')	0.5 SPICLK + 2VCLK	-	Delay between $\overline{\text{SPISCS}}$ assertion by the Master to the first edge of its SPICLK. This timing is based on CSHOLD being '0' for the current and the previous buffer tranfer
4b	$\overline{\text{SPISCS}}$ - SPICLK edge Phase = '1' (4pin configuration- Master, CSHOLD = '1')	0.5 SPICLK + 3VCLK	-	Delay between $\overline{\text{SPISCS}}$ assertion by the Master to the first edge of its SPICLK. This timing is based on CSHOLD being '1' for the current and the previous buffer tranfer
5	$\overline{\text{SPIENA}}$ - SPICLK edge Phase = '0' (4pin/5pin configuration - Master)	3VCLK	-	Delay between $\overline{\text{SPIENA}}$ assertion from the Slave to the start of first edge of SPICLK by the Master
6	$\overline{\text{SPIENA}}$ - SPICLK edge Phase = '1' (4pin/5pin configuration - Master)	0.5 SPICLK + 3VCLK	-	
7	SPICLK - $\overline{\text{SPIENA}}$ Phase,Polarity = "00" or "11" (4pin/5pin configuration - Slave)	-	1.5 VCLK to 2.5 VCLK	Delay between the last fall-edge of SPICLK to the deassertion of $\overline{\text{SPIENA}}$ pin by the Slave
8	SPICLK - $\overline{\text{SPIENA}}$ Phase,Polarity = "01" or "10" (4pin/5pin configuration - Slave)	-	1.5 VCLK to 2.5 VCLK	Delay between the last rise-edge of SPICLK to the deassertion of $\overline{\text{SPIENA}}$ pin by the Slave.
9	SPICLK - $\overline{\text{SPISCS}}$ Phase = '0', Master	0.5 SPICLK + 1VCLK	-	Delay between the last edge of SPICLK to de-activation of $\overline{\text{SPISCS}}$ pin by the Master
10	SPICLK - $\overline{\text{SPISCS}}$ Phase = '1', Master	1VCLK	-	
11	$\overline{\text{SPISCS}}$ - $\overline{\text{SPIENA}}$ (5pin), Slave	-	0	Delay between assertion of $\overline{\text{SPISCS}}$ by the Master to the assertion of $\overline{\text{SPIENA}}$ pin by the Slave (assuming Slave is ready with the TX data)

Table 11-52. Timing parameters of SPI/MibSPI pins.

SI No.	Delay Item	Master	Slave	Note
12	Hold time requirement on SPISOMI Input w.r.t SPICLK	0	-	The incoming data on SPISOMI pin is sampled w.r.t SPICLK itself
13	Hold time requirement on SPISIMO Input w.r.t SPICLK	-	0	The incoming data on SPISIMO pin is sampled w.r.t SPICLK itself
14	Maximum hold time on SPIENA after the final SPICLK to pin supported by Master (Phase = '0')	0.5 SPICLK + 1VCLK	-	After the last edge of SPICLK, the maximum delay within which SPIENA pin should be deasserted so that the Master does not start the next buffer transfer. SPIENA should be deasserted before this time.
15	Maximum hold time on SPIENA after the final SPICLK to pin supported by Master (Phase = '1')	1VCLK	-	After the last edge of SPICLK, the maximum delay within which nENA pin should be deasserted so that the Master does not start the next buffer transfer. SPIENA should be deasserted before this time.
16	Minimum delay between deselecting the Slave using its SPISCS pin and clocking a different Slave	-	1VCLK	A SPICLK edge within 1VCLK of deassertion of SPISCS will be treated a valid edge in Slave mode.
17	Minimum inactive time of SPISCS pin(s) between two data words	2VCLK	-	This can be increased by using WDELAY counter.
18	Time at which SPIENA pin will be sampled after SPISCS goes active	C2TDELAY	-	If C2TDELAY is not programmed or '0', then the FSM will look for the ENA pin value 1VCLK after the asserting the ChipSelect. The FSM always looks at the registered version of ENA pin value.
The ChipSelect to SPICLK or SPICLK to ChipSelect timings do not include C2T/T2C delay timer.				

All the numbers are logical delays. Actual delays between the pins will depend upon the differences in I/O buffer delays of each of the protocol pins. For example in a MASTER mode SPI, if the O/P buffer on SPICLK pin is of 4mA drive strength and the buffer on SPISIMO is of 2mA strength, then depending upon the loads on these pins, the delay difference between SPICLK & SPISIMO pins could vary between 5ns and 15ns or even more.

High-End Timer w/Parity (HET) Module

This section describes the high-end timer (HET) with parity support. The HET is a software-controlled timer with a dedicated specialized timer micromachine and a set of 21 instructions. The HET micromachine is connected to a port of input/output (I/O) pins.

	Topic	Page
12.1	Features	588
12.2	Overview	589
12.3	HET Functional Description	593
12.5	Angle Functions	620
12.6	HET Control Registers	624
12.7	Instruction Set	656

12.1 Features

The HET for the TMS470M Series device family has the following features:

- High-level timer functions such as period and pulse width measurement
- 32 input/output (I/O) channels for timer functions such as capture, compare, pulse width measurements (PWMs), and general purpose I/O pins
- 24 high resolution (HR) hardware channels associated with 24 of the 32 I/O channels
- 8 loop resolution (LR) hardware channels associated with 8 of the 32 I/O channels
- User-programmable loop resolution and HR clocks
- User-programmable micromachine with a reduced instruction set (21 instructions) for build-time functions
- Multiple 20-bit virtual counters for timers, event counters, and angle counters
- Dual port RAM with capacity for 64 words of 96 bits (expandable to 256 words, see device-specific datasheet) to optimize the number of cycles per instruction (most instructions require one cycle)
- Shadow registers for CPU communication
- Minimal CPU code overhead and required interrupts
- 35 interrupt sources with two individually programmable levels
- HR I/Os and coarse resolutions implemented by sub loops for multiple resolution capability
- Conditional program execution based on pin conditions and compares
- Software breakpoint capability on each instruction
- Synchronously operates with VCLK2
- Possibility of using multiple HETs synchronized on the same resolution clock
- Programmable pullup/pulldown select
- Programmable open-drain capability
- Digital and analog loopback mode

Note:
Refer to the device datasheet for the actual number and mix of HET channels.

12.2 Overview

The HET is a third-generation Texas Instruments (TI) advanced intelligent timer module.

This HET module provides sophisticated timing functions for real-time applications such as car engine management. The new HR 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 generic and comprehensive instructions, has improved the definition and development cycle time of an application and its derivatives. With the HET breakpoint feature combined with various stop capabilities, the TMSx70 HET is designed for easy software application debug.

12.2.1 Timer Module Structure and Execution

The timer consists of a specialized micromachine that operates a reduced instruction set at the same speed as VCLK2. Most of the data, arithmetic, and logical unit (ALU) are 25 bits wide. Two 20-bit registers and one 25-bit register 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 CPU to fetch the opcode and data in one VCLK2 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 includes an 8-bit field for specifying the address of the next instruction to be executed. This means that an application program sequence is not controlled by a program counter (PC), but by the actual content of each instruction. This arrangement offers greater flexibility in going back and forth in the memory during program execution.

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 after reset before the timer starts execution. Once the timer program is loaded into the memory, the CPU starts the timer execution, and only data parameters can then be 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 instruction sequence implements a timer. A PWM would need a two-instruction sequence: count and compare. A complex time function may include many instructions in the sequence. The total timer program is a set of time functions executed sequentially, one after the other. Reaching the end, the program must roll to the first function so that it behaves as a loop.

The time for a loop to execute is referred to as a *loop resolution clock cycle*. When the HET rolls over to the first function (i.e., instruction), the timer waits for the resolution clock to restart the execution of the loop to ensure that only one loop is executed for each loop resolution clock cycle.

The execution time of this main loop is the sum of all the cycles required for the instructions that the loop executes. The main loop must be completed within the loop resolution clock. Therefore, you must make sure that the timer functions implemented in the algorithm are executed within the resolution clock. Otherwise, the program will behave unpredictably because some instructions will not be executed each time through the loop.

This requirement creates a strong link between the accuracy of the timer functions and the number of functions (the number of instructions) the timer can perform. To overcome this limitation, some of the most commonly used instructions can access the HR hardware. This access allows pulse or period measurements, time compare, and PWM output waveforms at the resolution of the HR clock instead of the loop resolution clock.

Updating parameters of compare instructions (time, event, or angle compare values) can be controlled by the timer itself using built-in move instructions. These built-in move instructions actually transfer new data from the CPU into the active compare field of an instruction synchronously with the resolution clock. This synchronization method makes it unnecessary to use interrupts to avoid such problems as incorrect pulse widths.

12.2.2 Major Advantages

In addition to classic time functions such as input capture or multiple independent 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 all the flow control inside the HET program itself. More complex algorithms can take advantage of the CPU access to the HET 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 HET RAM. CPU access to the HET RAM also improves the debug and development of timer programs. The CPU program can stop the HET and view the contents of the program, control, and data fields that reside in the HET 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 24 HR channels. The standard memory structure allows module configuration from 64 words to 256 words of timer program memory.

12.2.3 Performance

Most instructions execute in one cycle, but a few take two or three cycles. The average cycle per instruction (CPI) measured on various complex benchmarks is approximately 1.3.

The HET can generate many complex output waveforms without CPU interrupts. Where special algorithms are needed following a specific event (e.g., missing teeth or a short/long input signal), a minimal number of interrupts to the CPU are needed. The minimal interrupts frees the CPU bandwidth to perform other tasks.

12.2.4 Instructions Features

The TMS470M Series HET has the following instructions features:

- All standard resolution outputs are synchronized to the loop resolution clock to avoid the variability associated with the instruction placement in the HET program.
- HET uses a RISC-based specialized timer micromachine to carry out a set of 21 instructions.
- Instructions implemented in a VLIW format (96 bits wide).
- The HET program execution is self-driven by external or internal events, branching to special routines based on input edges or output compares.

12.2.5 Parity Support

- A Parity support for HET memories (program, data, and control) has been implemented. A parity bit will be calculated on a HET word while writing to the HET RAM and stored as a single-bit in the Parity region of the HET RAM.
- Parity checking is done for even or odd parity on the RAM word based on the system module configuration.
- Parity checking can be enabled/disabled within the HET itself by programming a 4-bit key in the HET Parity Control register. During reads, parity is re-calculated on the HET word and compared against the parity already stored in the RAM.
- In case of parity error:
  - HET generates an interrupt (HET_UERR_INT) to the VIM or ESM if it is enabled.
  - HET will be switched off at the loop resolution clock boundary, if the HET_STOP bit is programmed to '1' (this is done by programming the appropriate bit in the HETGCR register).
  - The RAM address for which the parity error is detected will be captured in a register for debugging purposes. This address is frozen from being updated until the bus master reads it.
  - During Debug mode (SUSPEND = 1) a parity error will not cause an interrupt, as the HET will continue to execute and the Parity Address register (HETPAR) will not be updated.

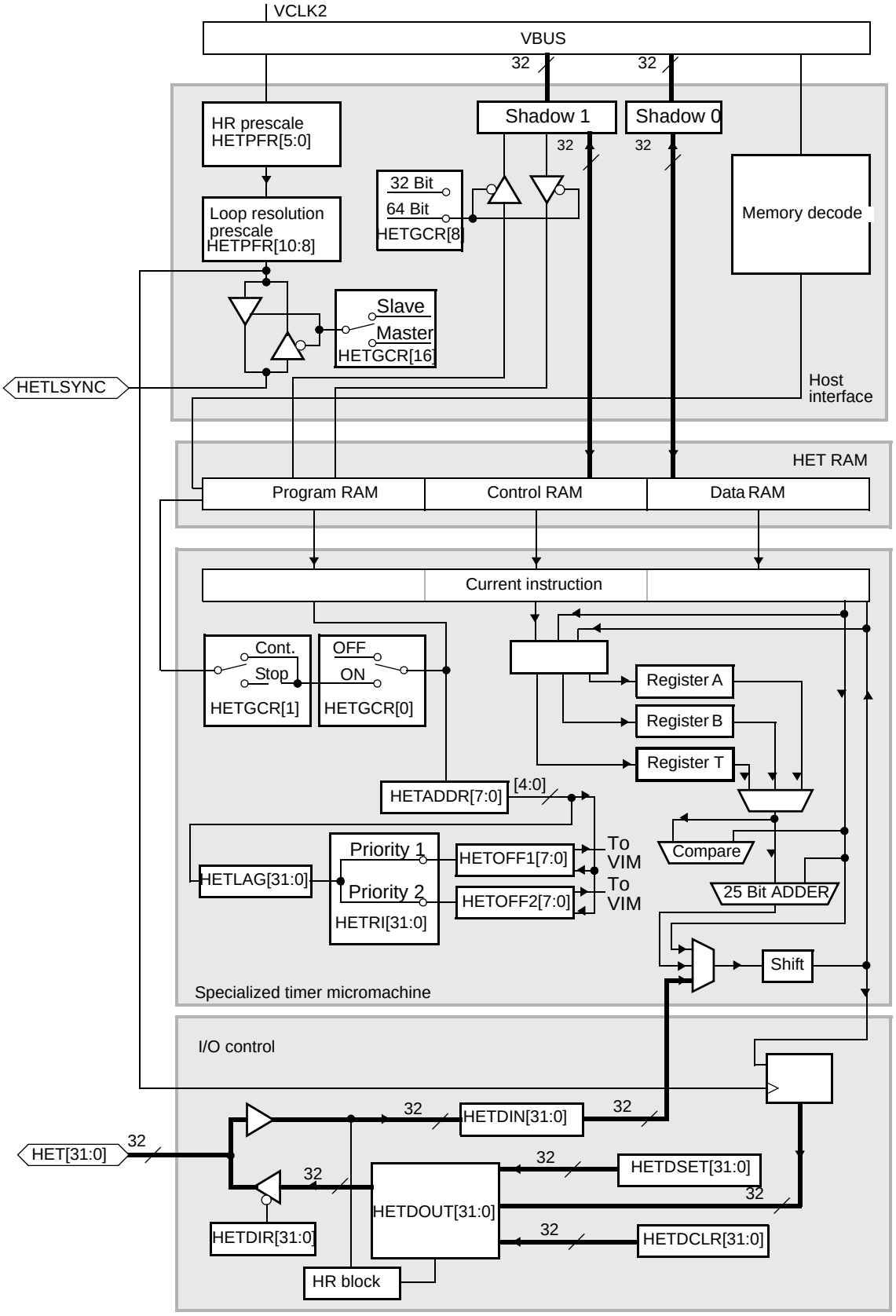
12.2.6 Block Diagram

The HET module (see Figure 12-1) comprises four separate components:

1. Host interface
2. HET RAM
3. Specialized timer micromachine
4. I/O control

The HET is attached to an I/O port of up to 32 pins. Please see the device-specific data sheet for details on the number of HET pins available.

Figure 12-1. HET Block Diagram



12.3 HET Functional Description

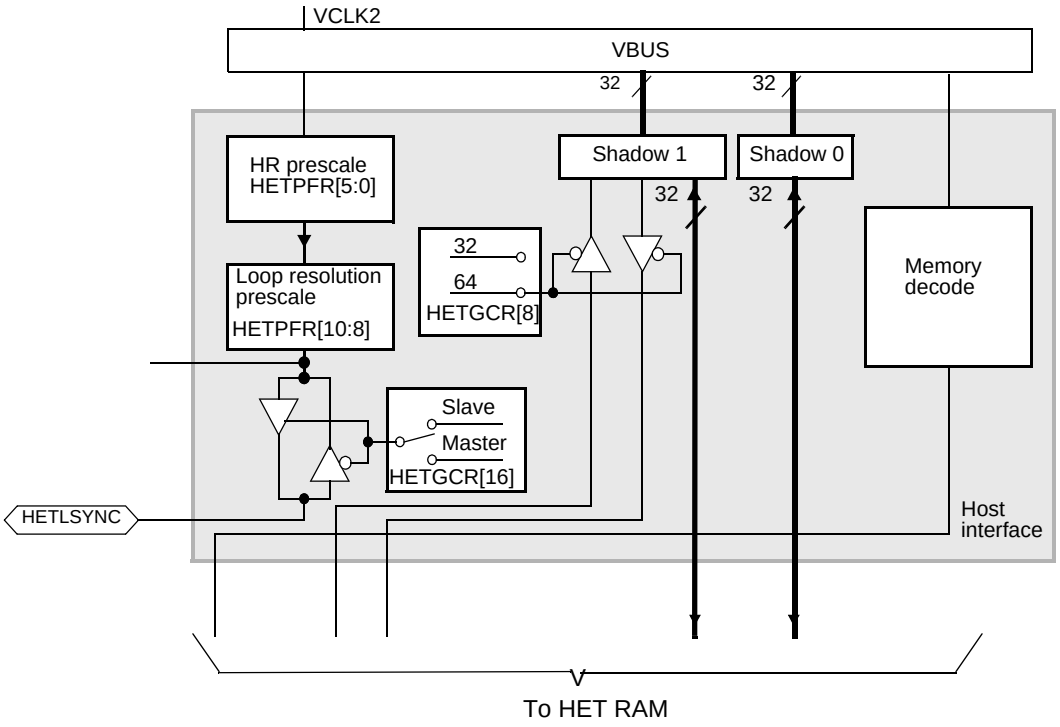
The HET contains a host interface to the CPU and RAM into which HET code is loaded. The HET code is executed by the specialized timer micromachine. The I/O control provides an interface to external pins respectively.

12.3.1 Host Interface

The host interface controls all communications between timer-ram and CPU (see Figure 12-2). It includes the following components:

- Two 32-bit shadow registers (shadow register 0 and 1)
- A bit (HETGCR[8]; Section 12.6.1) for controlling how the shadow registers are written/read by the CPU
- Two user programmable 6-bit(HETPFR[5:0]; Section 12.6.2) and 3-bit (HETPFR[10:8]) clock pre scalers for HR and loop resolution clocks
- A bit (HETGCR[16]; Section 12.6.1) for controlling whether the HET is configured as a master or a slave

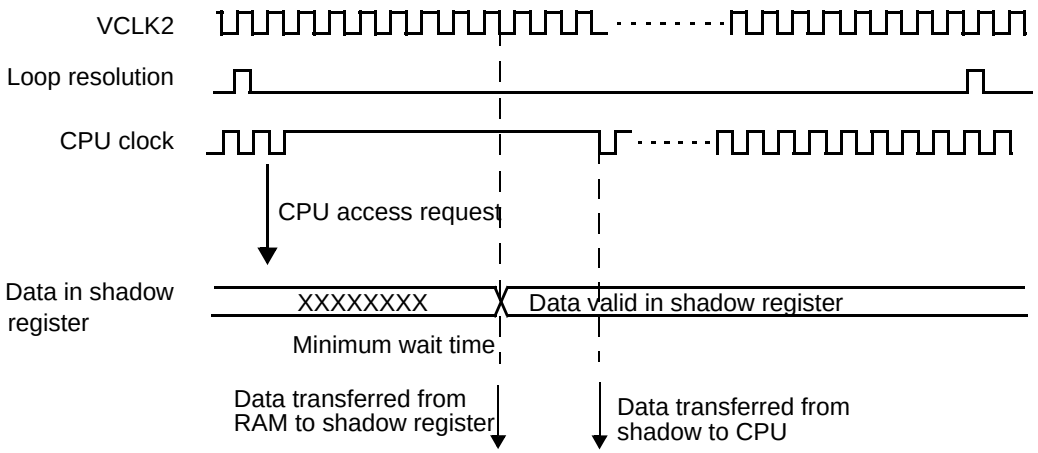
Figure 12-2. Host Interface



12.3.1.1 Memory Control

Each CPU access to the HET RAM will take seven VCLK2 cycles in addition to the cycles needed on the peripheral bus, which will be dependent on the HCLK to VCLK ratio programmed in the device.

Figure 12-3. HET RAM Accesses (Example of a 10-Wait Cycle Read Operation)



Each time the CPU performs an access to the HET RAM, one or two cycles of the HET program are dedicated to the CPU access since the CPU and the HET cannot access the HET RAM at the same time. A 32-bit read takes one cycle and a 64-bit takes two cycles. One of the most common uses of a 64-bit read is for the PCNT instruction. The 64-bit read gives you the control field (previous period) and data field (current period) at the same instant in time. The total number of available time slots for the HET must be computed by subtracting the time slots consumed by the CPU from the number of cycles available in a loop resolution. In the worst case of continual CPU access to the HET, the HET loses one time slot (or two cycles in case of auto read/clear of PCNT) out of every eight.

Example:

An example HET configuration allows 24 time slots in a loop resolution (by the prescalers). The memory controller is programmed with seven wait-state to the HET. In the case of continuous CPU access, three cycles per resolution will be dedicated to the CPU, leaving 21 time slots really available for the HET application program.

12.3.1.2 Shadow Registers

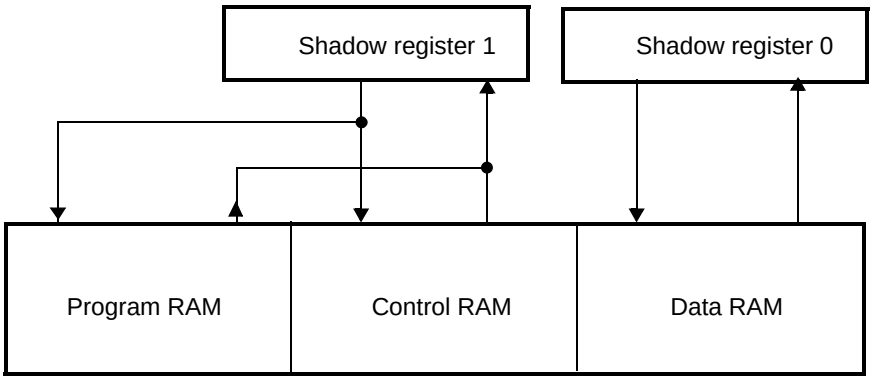
The two shadow registers allow the CPU to access the program, control, or data field in the timer-RAM without interrupting execution of a timer instruction. Shadow register operations are transparent to you. See Figure 12-4.

Shadow registers allow the timer instructions to access the timer control field and data field 64 bits at a time. To be active, this 64-bit mode must be set in the global configuration register.

The timer-RAM cannot be accessed by byte or half-word (16-bit data).

- Shadow register 0 is assigned to data field access.
- Shadow register 1 shares program field and control field accesses.

Figure 12-4. Shadow Registers with RAM Units



12.3.1.3 CPU Access to Timer-RAM

- 32-bit access (HETGCR[8] = 0; Section 12.6.1) (program, control and data field)
  - **Write**: When the CPU executes STR, it first writes 32-bit CPU data into relevant shadow registers. The HET asserts then 7 additional VCLK2 wait cycles to the R2V. The CPU resumes its execution due to the write buffer implemented in the R2V. Should another HET access occur while the waitstates are active, the CPU will be put into waitstate for the remaining wait cycles. At the end of the current timer instruction, the shadowed data moves into timer selected RAM.
  - **Read**: When the CPU executes LDR, the HET asserts then 7 additional VCLK2 wait cycles to the CPU. At the end of the current instruction, the 32-bit timer data are loaded into relevant shadow registers. At the end of the wait time, the shadowed data moves onto the data bus.
- 64-bit access (HETGCR[8] = 1; Section 12.6.1)
  - **Write**: For a 64-bit write, two STR CPU instructions are required. The control field must be written first, followed by the data field. For the first STR instruction, the CPU writes 32-bit CPU data into shadow register 1. The HET asserts then 7 additional VCLK2 wait cycles to the R2V, but the CPU can still remain executing the next instructions. For the second STR instruction, the CPU writes 32-bit CPU data into shadow register 0. The CPU will be put into waitstates, should the previous wait cycles still be active. The HET asserts then 7 additional VCLK2 wait cycles to the CPU for the second access. At the end of the current timer instruction, all 64-bit shadowed data moves into timer RAM.
  - **Read**: For a 64-bit read, two LDR CPU instructions are executed. The control field must be read first, followed by the data field. For the first LDR instruction, the HET asserts then 7 additional VCLK2 wait cycles to the CPU. At the end of the current timer instruction, the timer loads 32-bit control field into shadow register 1 and the 32-bit data field into shadow register 0 (both in the same cycle). At the end of the wait time, the shadowed data from the control field move onto the data bus. For the second LDR instruction, the HET asserts then 7 additional VCLK2 wait cycles to the CPU. The CPU then moves the 32-bit of shadow register 0 (loaded from the data field at the first LDR) onto the data bus.

Note:
Interrupts:
During the time the 64-bit access bit is set, it is recommended to disable all interrupts.
Background**:**
- Avoid an interrupt routine that causes a delay between the first and the second LDR instruction.
- Disabling the interrupts is a must if an interrupt routine also reads the HET RAM locations.

- Automatic read/clear of the HET RAM data field

The HET provides a feature allowing to automatically clear the data field after a 64-bit read operation of the control and data fields. This feature is implemented via the control bit, which is located in the control field (bit C21). This is a static bit that can be used by any instruction.

1. Set the control bit in the selected instruction.
2. Set the 64-bit access bit in the global configuration register.
3. Perform a 64-bit read access with the CPU.
4. Loads shadow registers 1 and 0 with the control and data fields of the memory, and then clears the data field [D24:D0] of the instruction. This sequence requires two time slots.

This feature is mainly used for PCNT instruction. You read a period or pulse value, which is then cleared by the HET itself. This means that you do not need to take a new value as long as the period or pulse data is still cleared.

12.3.1.4 Memory Selects

The start address of the HET RAM is device dependent. Please see the device specific datasheet for further information.

12.3.1.5 Emulation Mode

Emulation mode, used by the software debugger, is specified in the global configuration register. When the 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. Wait states asserted by the timer while the suspend signal is active are executed as normal.

12.3.1.6 Power-Down

The HET 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 HET. During global low power mode, all clocks to the HET are turned off so the module is completely inactive.

Local low power mode can be asserted in two ways. One possibility is by setting the POWERDOWN (HETGCR [24]; Section 12.6.1) bit; setting this bit stops the clocks to the HET internal logic (state machine), but the HET registers continue to be clocked. Another way to stop the clock to the HET state machine and the registers is to put the peripheral select the HET is connected to into power down mode (write 1 to corresponding PSPWRDWNSETx register bit).

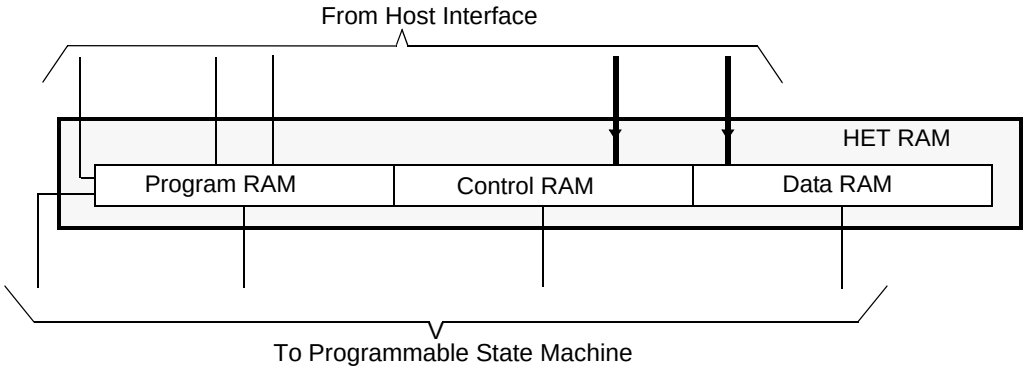
Exiting local power mode can be achieved by resetting to 0 the POWERDOWN (HETGCR [24]; Section 12.6.1) bit, to enable the clocks of the HET internal logic (state machine) or by writing a 1 to the appropriate PSPWRDWNCLRx registers (Section 12.6.1) if this scheme was used for putting the module into power down mode.

12.3.2 HET RAM

The timer RAM uses dual port access. This means that one RAM address may be written while another address is read. This feature is especially useful for the HET architecture because the prefetch is done during the last cycle of each instruction and the timer RAM allows writing back data into the current instruction while reading the next instruction. The RAM words are 96 bits wide, which are split into three 32-bit fields (program, control, and data) to fit with the CPU data bus. See Figure 12-5.

Note: RAM
The RAM supports word accesses only.

Figure 12-5. HET RAM



12.3.2.1 Memory Map

Table 12-1 describes the HET memory map.

Table 12-1. HET Memory Map

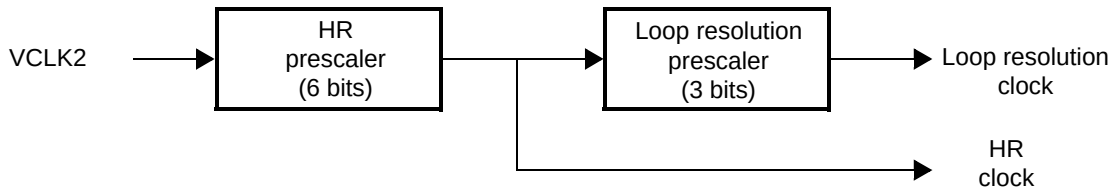
Instruction Address	Program Field Address	Control Field Address	Data Field Address	Reserved Address
00h	XX000h	XX004h	XX008h	XX00Ch
01h	XX010h	XX014h	XX018h	XX01Ch
02h	XX020h	XX024h	XX028h	XX02Ch
:	:	:	:	:
:	:	:	:	:
3Fh	XX3F0h	XX3F4h	XX3F8h	XX3FCh
40h	XX400h	XX404h	XX408h	XX40Ch
:	:	:	:	:
FFh	XXFF0h	XXFF4h	XXFF8h	XXFFCh

12.3.3 Time Base

Two prescalers generate the timer resolution clock for the program loop and the HR clock for the HR I/O counters. See Figure 12-6. The prescalers consist of the following:

- A 6-bit prescaler dividing VCLK2 by a user-defined HR prescale divide rate (hr) stored in the 6-bit HR prescale factor code (with a linear increment of codes). See Table 12-2.
- 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 (with a power of 2 increment of codes). See Table 12-3.

Figure 12-6. Prescaler Configuration



The LR is typically determined by the number of cycles that the HET requires to complete the worst-case application software loop plus CPU accesses. Therefore, you must choose the prescaler numbers so that the number of time slots in any software loop is greater than the number of cycles required in a loop. Typically, you choose the smallest HR prescaler number that will still allow the appropriate number of cycles.

- Time slots available (Ts)

  Ts = [HR prescale divide rate (Hr) * loop resolution prescale divide rate (Lr)] -1

- HR clock period (HRP)

  *HRP = Hr * VCLK2 period* (Tclk) = Hr / VCLK2

- Loop resolution clock period (LRP)

  LRP = loop resolution prescale divide rate (LR) * HR clock period (HRP) = Lr * Hr *Tclk = (Hr * Lr)/ VCLK2

Table 12-2. HR Prescale Factor Codes (Continued)

HR Prescale Factor Code						HR Prescale Divide Rate
5	4	3	2	1	0	
0	0	0	0	0	0	1
0	0	0	0	0	1	2
0	0	0	0	1	0	3
:	:	:	:	:	:	:
:	:	:	:	:	:	:
1	1	1	1	0	1	62
1	1	1	1	1	0	63
1	1	1	1	1	1	64

Table 12-3. Loop Resolution Prescale Factor Codes

Loop Resolution Prescale Factor Code				Loop-Resolution Prescale Divide Rate
ordinal	2	1	0	
0	0	0	0	1
1	0	0	1	2
2	0	1	0	4
3	0	1	1	8
4	1	0	0	16
5	1	0	1	32
6	1	1	0	reserved
7	1	1	1	reserved

Note:
When the loop-resolution prescale divide rate is smaller than 32, the non-relevant bits of the HR data fields (non-relevant LSBs) will be one of the following; see also Table 12-4:

•Written as 0 for HR capture

•Or interpreted as 0 for HR compare

For more information about this, see the descriptions of the ECMP (Section 12.7.3.10), MCMP (Section 12.7.3.12), PCNT (Section 12.7.3.15), PWCNT (Section 12.7.3.16), and WCAP (Section 12.7.3.21) instructions.

Table 12-4. Interpretation of the 5-Bit HR Data Field

Loop Resolution Prescale divide rate (Lr)	Bits of the HR data field[1]					HRP Cycles delay range
	• B[4]	• B[3]	• B[2]	• B[1]	• B[0]	
1	X	X	X	X	X	0
2	V	X	X	X	X	0 to 1
4	V	V	X	X	X	0 to 3
8	V	V	V	X	X	0 to 7
16	V	V	V	V	X	0 to 15
32	V	V	V	V	V	0 to 31
Weight factor as fraction of HR Cycles in one loop	1/2	1/4	1/8	1/16	1/32	

1 V = Valid bit; X = Non-relevant bit (ignored)

Example:

HETPFR[31:0] register = 0x300 → Lr prescale divide rate = Lr = 8 (→ 8 Hr cycles in one loop).

Assumption: HR data field = 0x14 = 10100b

Lr = 8 → Bits B[1] and B[0] are ignored → Hr delay = 101b = 5 Hr Cycles

or by using the calculation with weight factors:

Hr Delay = Lr * (B[4] * 1/2 + B[3] * 1/4 + B[2] * 1/8 + B[1] * 1/16 + B[0] * 1/32)
= 8 * (1 * 1/2 + 0 * 1/4 + 1 * 1/8)
= 5 Hr cycles

12.3.3.1 Determining Loop Resolution

As an example, consider an application that requires HR I/Os with a resolution of 62.5 ns, and 8 standard IOs at 2 μs, and needs 60 time slots for HET application programs.

Considering a VCLK2 frequency of 32 MHz, the following divide-by rates can be programmed using the following prescaler factor codes:

- Divide-by rate of 2 for the HR clock, giving 62.5 ns (code = 000001 = 0x1):
  HRP = Hr / VCLK2 = 2 /32 Mhz = 62.5 ns
- Divide-by rate of 32 for the loop resolution clock, giving 2 μs (code = 101 = 0x5)
  LRP = Lr * HRP = 32 * 62.5 ns = 2 us
- Number of available time slots: Ts = (Hr * Lr) -1 = (2 * 32)-1 = 63
- Hr =2, Lr = 32. Then, the corresponding code inside the HETPFR[31:0] register (Section 12.6.2) will be: 0x00000501

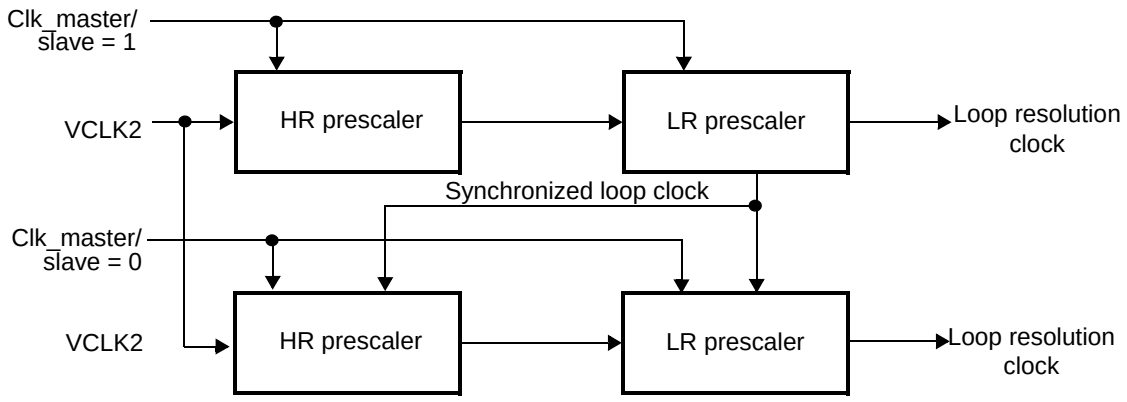
If in the example above, the resolution needed moves from 2 μs to 1 μs, then only 31 time slots will be available.

Or, if the number of time slots needed are 90, HRP then moves to 93.75 ns.

12.3.3.2 Multi-HETs Resolution Synchronization

In some applications configured with multiple HETs, the resolutions must be synchronized. This is typically the case for configurations that use a unique angle reference shared by several HETs. See Figure 12-7.

Figure 12-7. Multi-HETs Configuration—Synchronization of Resolutions



Note: Synchronization
The loop clock signal is two VCLK2 cycles ahead of the resolution clock.

The HET provides a synchronization mechanism for multiple timers. The Clk_master/slave (HETGCR[16]; Section 12.6.1) configures the HET in master or slave mode (default is slave mode). A HET in master mode provides a signal to synchronize the prescalers of the slave HET. The slave HET synchronizes its loop resolution to the loop resolution signal sent by the master. The slave does not require this signal after it receives the first synchronization signal. However, any time the slave receives the resynchronization signal from the master, the slave must resynchronize.

Follow these steps to synchronize multiple HETs.

1. Select one HET as master by writing in the clk_master/slave bit.
2. Configure the HR and LR prescaler control registers and all other control registers (direction, interrupts, etc.) of the master HET.
3. Configure the HR and LR prescaler control registers and all other control registers (direction, interrupts, etc.) of the slave HET. The loop resolution of the master **must be the same** as the slave's loop resolution.
4. Turn on all slave HETs. They will wait for the synchronization from the master HET to start the program execution.
5. Turn on the master HET.

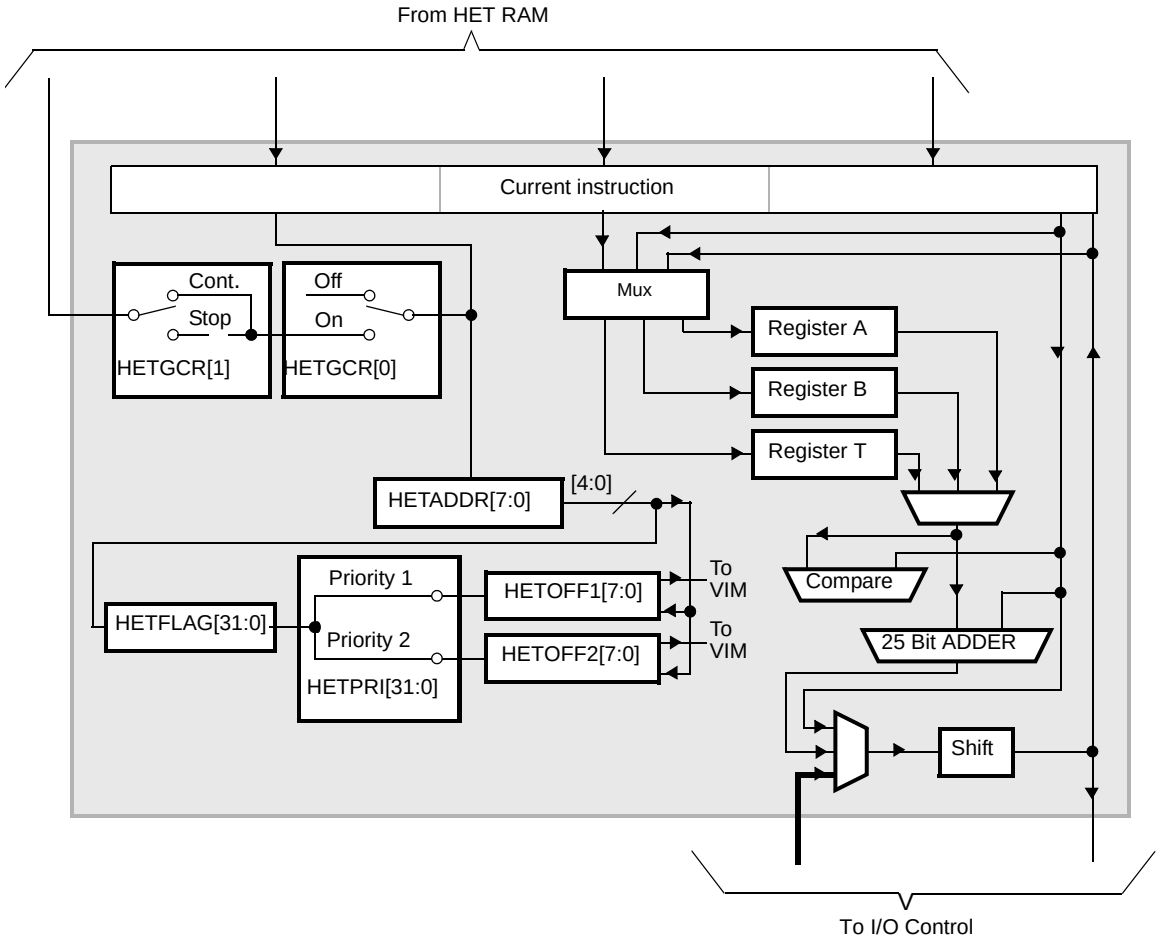
12.3.4 Specialized Timer Micromachine

The HET has its own instruction set, detailed in Section 12.6.6. The timer micromachine reads each instruction from the HET 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 HET RAM sequentially. The HET 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 an indexed addressing mode.

Figure 12-8 shows the major operations that the HET can carry out: compares, captures, angle functions, additions, and shifts. The HET is also capable of special operations, which the user can construct using the major operations. The HET contains three registers (A, B, and T) used to hold compare or counter values to be used by the HET 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.

Figure 12-8. Specialized Timer Micromachine



12.3.4.1 Time Slots and Resolution Loop

Each instruction requires a specific number of cycles or time slots to execute. The resolution specified in the prescaler registers 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 VCLK2 accuracy) on the HR I/O pins. For more information, see Section 12.3.5.

12.3.4.2 Program Loop Time

The program loop time is the sum of all cycles used for instruction execution and CPU accesses. This time may vary from one loop to another depending on the number of CPU accesses or special routine execution.

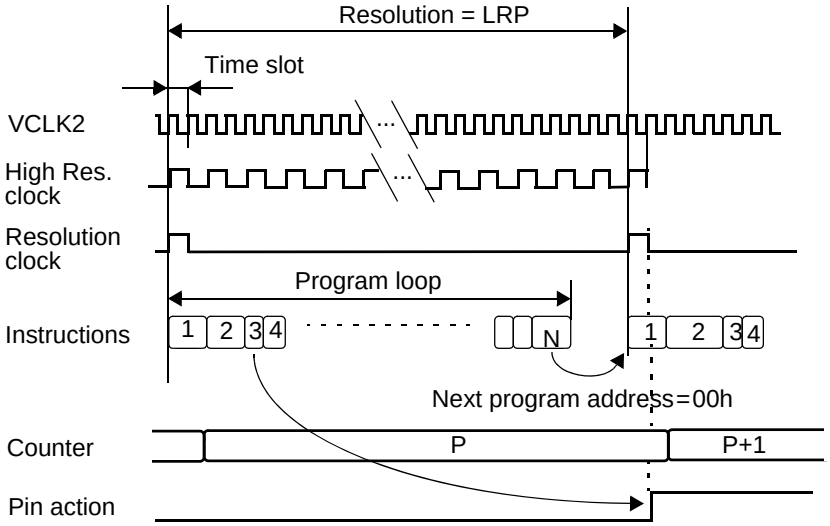
The timer program restarts on every resolution loop. The start address is fixed at RAM **address 00h**. The longest loop in a program must fit within one resolution loop time to guarantee complete accuracy.

The last instruction of a program points to the start address, (next program address= 00h).

If a program loop is shorter than the selected resolution, the program waits until end of resolution before starting the next loop.

The timing diagram (see Figure 12-9) illustrates the program flow execution.

Figure 12-9. Program Flow Timings

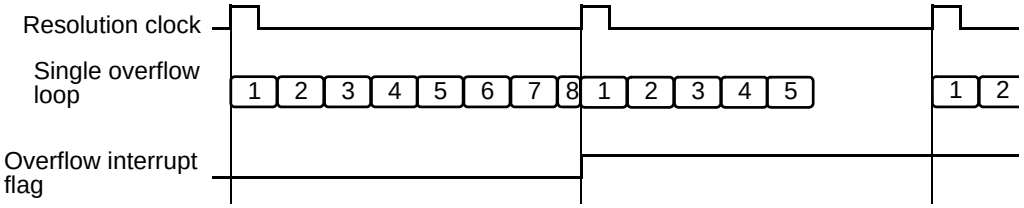


12.3.4.3 Program Overflow

If the number of time slots used in a program loop exceeds the number available in one resolution, the timer sets the program overflow interrupt flag located in the exceptions interrupt control register. To maintain synchronization of the I/Os, this condition should never occur in a normal operation. During the software debug phase, this flag may be used to debug the programmed loop resolution value. See Figure 12-10.

In case of program overflow, the HET sequence is started again at address 00h.

Figure 12-10. Use of the Overflow Interrupt Flag



Note: Limitation on instruction falling on a program overflow
The pin action will not be taken if the instruction that caused the pin action falls on a program overflow. All other instructions such as updating the A, B, and T registers and the RAM will still be performed.

12.3.4.4 Instruction Execution Sequence

The execution of a HET program begins with the first occurrence of the loop resolution clock after the HET 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 HET goes into a wait state until the occurrence of the loop resolution clock and resumes normal execution. If a IDLE state is asserted (i.e., CPU access) at the beginning of the program, (at the occurrence of the loop resolution clock), the execution of the program is postponed by one or two VCLK2 cycles.

Single and multi-cycle instructions prefetch the next instruction in the last cycle of the instruction. SCNT, the only three cycles instruction (see Section 12.7.3.19), fetches the next instruction always in the third cycle, which is the last cycle of the instruction. ECMP, as other single cycle instruction (see Section 12.7.3.10),

fetches the next instruction while executing the current instruction. By prefetching the instruction at the next or conditional address, the instruction is latched by the instruction registers and decoded in the same cycle.

The use of a dual-ported RAM permits dual access to the instruction memory. A single read and a single write can be performed in the VCLK2 cycle. In fact, a read and write to the same location can also be performed in the same VCLK2 cycle. The write operation is performed, regardless of the prefetching of instruction, in the cycle required by the instruction. Single cycle instruction can perform the following actions in one cycle:

1. Latch the current instruction.
2. Decode and execute the latched instruction.
3. Perform a write instruction to the instruction RAM.
4. Prefetch the instruction at the next or conditional address.

Multi-cycle instructions perform step 1 and 2 in the first cycle. Step 3 is executed in the cycle required by the instruction. Some instructions will perform multiple writes to the RAM, although in different cycles. Step 4 above is always performed in the last cycle of the instruction.

The number of instructions executed by a program must be between two loop resolution clocks. The program executes its first instruction in the high phase of the loop resolution clock and stops the execution of the program before the occurrence of the LClock_Last. If the total time of execution for the program exceeds the limit, the program overflow flag is asserted and the execution of the program is stopped. The execution of the program is resumed by prefetching the instruction at location address 00h.

Restrictions

Note:
The size of a HET program should be greater than one instruction. If there is only one instruction, this instruction will be executed twice.

Note:
Any MOVE instruction (see Section 12.7.3.13 and Section 12.7.3.14) that modifies a field in the next instruction to be executed will incur a wait cycle.

Note:
HET instruction decoding goes out of sync when a ADCNST or MOV instruction is having the remote address equal to next address. Such a scenario shall be avoided in application. Otherwise, the prefetch (of next address) may occur before the data is written to next address.

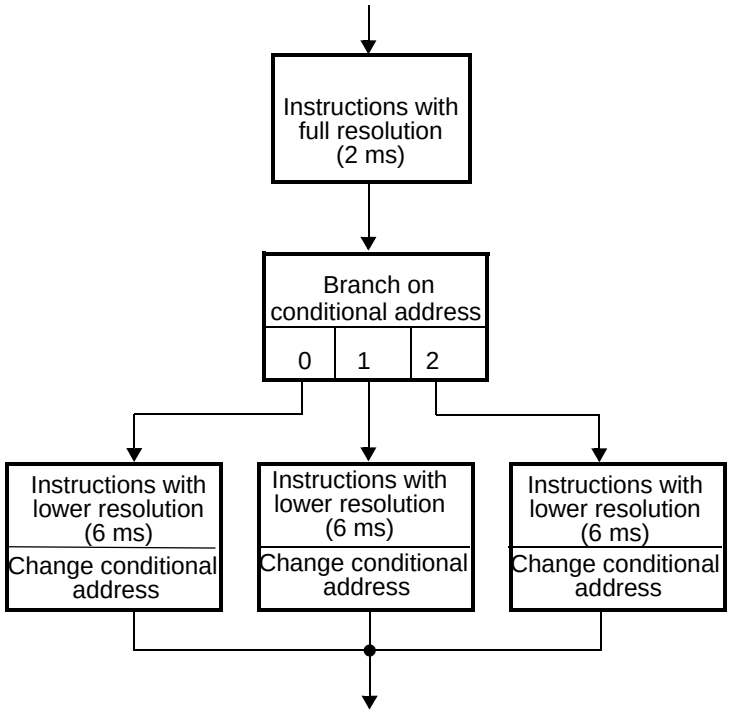
12.3.4.5 Multi-Resolution Scheme

When full resolution is not required, special program sequences with lower resolution can be used to optimize the use of time slots.

A lower-resolution sequence consists of instructions that are not executed on every resolution loop but only on every *n* resolutions. Unconditional branch instructions and an index sequence, using a MOV64 instruction (see Section 12.7.3.14) in each low resolution loop, is required to control this particular program flow. See Figure 12-11.

Note:
HR instructions must be placed in the main (full resolution) loop to ensure proper operation.

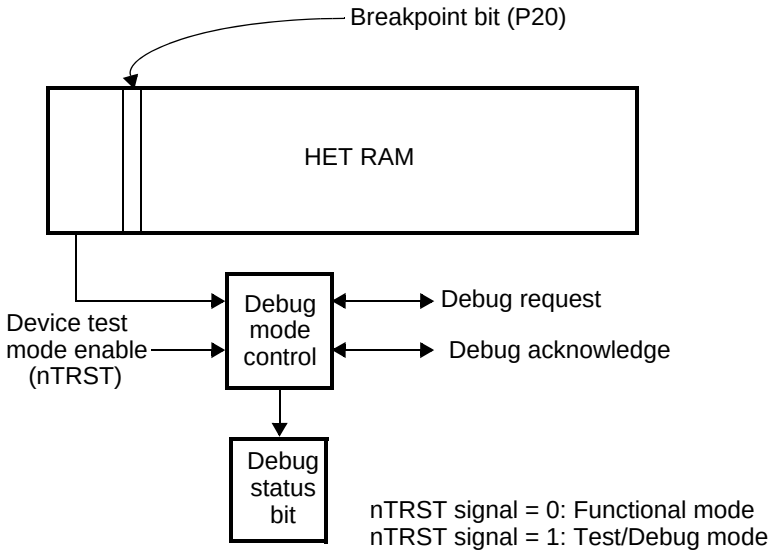
Figure 12-11. Multi-Resolution Operation Flow



12.3.4.6 Debug Capability

The HET has built-in debug capability in its architecture that allows you to more easily debug your HET program. See Figure 12-12.

Figure 12-12. Debug Control Configuration



This debug capability uses the breakpoint bit that is included as part of each HET instruction (bit P20 of any instruction). This bit can be programmed to a 1 to cause the HET program to freeze after the current instruction is executed. This action will also cause all internal HET state machines to freeze and send a debug request to the CPU. During the HET debug (i.e., HET breakpoint reached), you may still access the contents of all HET control registers. This includes a register that contains the current HET address. This allows you to determine which instruction caused the breakpoint condition. If the breakpoint bit is then programmed to a 0, the instruction will then be executed and resume program operation.

When the device test mode is enabled and a breakpoint is reached in the HET program, the debug status bit (HETGCR[2]) will be set and the debug request signal will be sent to the CPU. The CPU will wait for the CPU instruction boundary and then send the debug acknowledge signal back to the HET. At this time the debugger may clear the debug status bit by writing a 1 to clear the bit in the HET peripheral frame. This will allow the HET to exit its debug mode when the debug acknowledge signal is deasserted by the CPU.

For example, an equality compare (ECMP) instruction could be used to branch to an instruction with the breakpoint bit set when the compare is equal. A software capture word (WCAP) instruction could be used to branch to an instruction with the breakpoint bit set when a certain external pin condition exists. When a breakpoint instruction is executed, the HET freezes and the CPU enters debug mode.

The HET also supports the direct assertion of debug mode from the CPU. If the CPU goes into debug mode, it will signal this to the HET. If the ignore suspend bit is not set, the HET freezes all state machines and allows CPU accesses just as it does during the HET-initiated debug mode described above. If the ignore suspend bit is set, the HET will resume its execution, although the CPU is in debug mode.

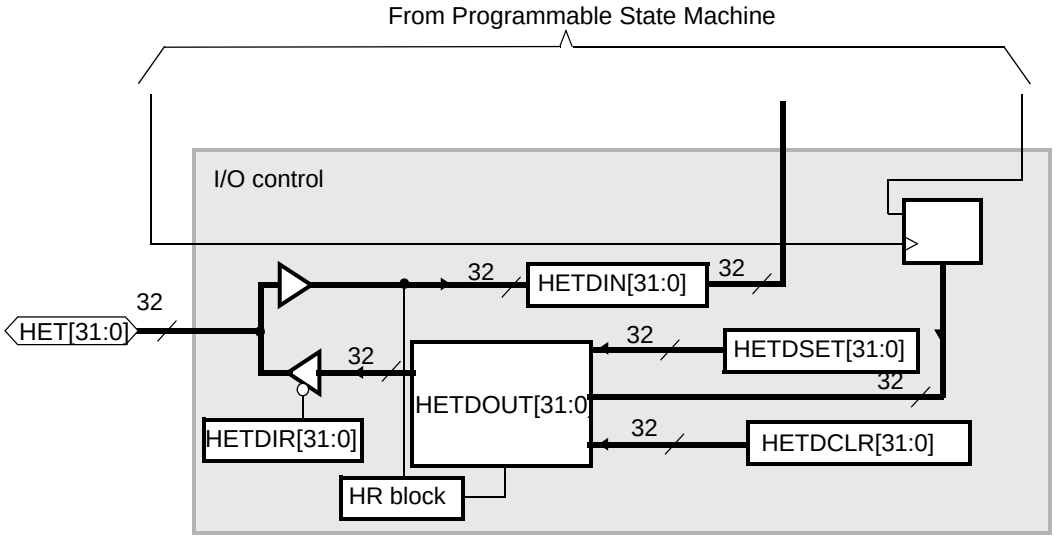
12.3.5 I/O Control

The HET has a port capable of up to 32 pins, but not all might be available on each device. All the HET pins available are programmable as either inputs or outputs.

Note:
Refer to the device configuration to determined the implemented pins.

The 32 I/Os are identically structured and are connected to pins HET[31] to HET[0]. This structure allows any HET instruction to use these I/Os with a loop resolution accuracy. See Figure 12-13 for an illustration of the I/O control. Among these 32 I/Os, 24 have an additional HR structure (pins HET[23] to HET[0]) based on the HR clock.

Figure 12-13. I/O Control



The CPU can use general purpose I/Os, pins HET [31] to HET [0], for reading using the control register HETDIN (Section 12.6.13), and for writing using HETDOUT (Section 12.6.14), HETDSET (Section 12.6.15), or HETDCLR (Section 12.6.16).

Note: Using HETDSET and HETDCLR
The HETDCLR and HETDSET registers (Section 12.6.16 and Section 12.6.15) can be used to minimize the number of accesses to the peripheral to modify the output register and output pins. Actually, when the application needs to set or to reset *n* pins without changing the value of the

other pins, the first possibility is to read HETDOUT, modify the content (AND, OR, etc.), and write the result into HETDOUT. This solution will take, in the best case, eight Cycles (LDR,AND/OR, STR), and could be interrupted by a function modifying the same register, which will result in a data coherency problem.

Using the HETDSET or HETDCLR registers, 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. This operation will take only three cycles (STR) and is not interruptible.

Example (C program): Set pins using the two 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 */
```

12.3.5.1 Loop Resolution Structure

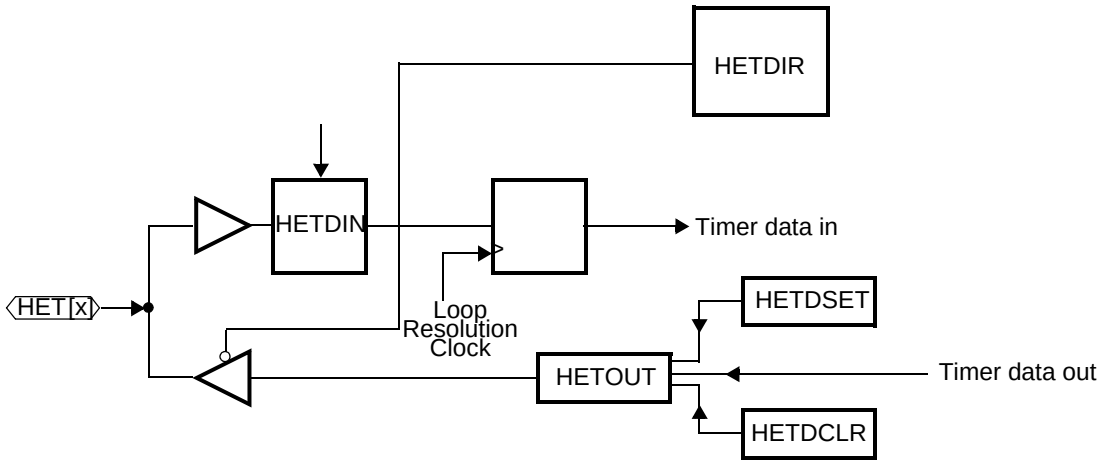
The HET uses the pins HET [31:0] as inputs and/or outputs. By using the I/O register of the HET, the CPU can interact with the HET program flow.

When an action (set or reset) is taken on a pin by the HET program, the HET will modify the pin at the rising edge of the next resolution clock.

When an event occurs on a HET I/O pin, it is taken into account at the next rising edge of the resolution clock.

The structure of each pin is shown in Figure 12-14.

Figure 12-14. HET Loop Resolution Structure for Each Bit



12.3.5.2 HR Structure

Among the 32 I/Os, 24 have the special HR structure (pins HET [23:0]) based on the HR clock. See Figure 12-15. The HR clock frequency is programmed through register HETPFR[5:0]. See Section 12.6.2.

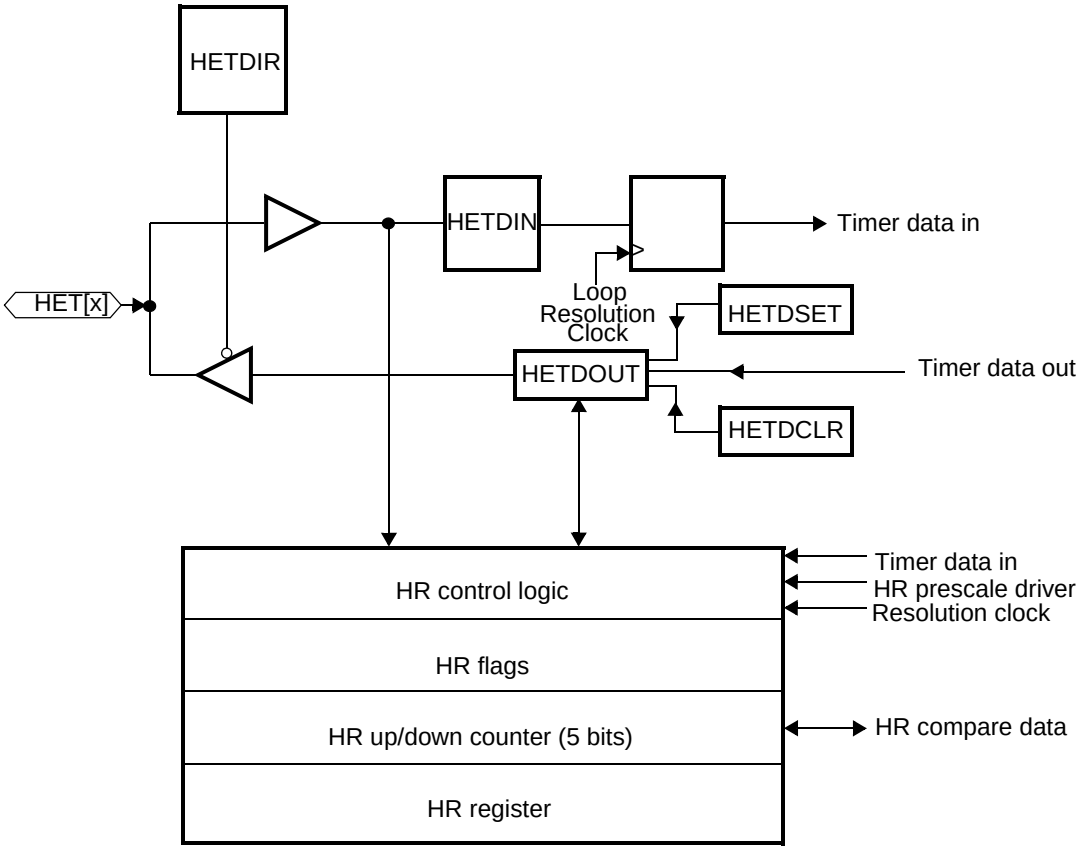
In addition to the standard I/O structure pins, HET [23:0] have special HR hardware so that these pins can be used as HR input captures (using PCNT or WCAP) or HR output compares (using ECMP, MCMP or PWCNT).

12.3.5.3 HR Block Diagram

Each time an HR instruction is executed on a given pin, the HR block for that pin is programmed (which HR function to perform and on which edges it should take an action) with the information in 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:
Only one HR function is allowed per resolution structure. To enforce this restriction, the architecture shown below is programmed only once per resolution loop. This occurs when the first instruction is executed. You must ensure that only one instruction accesses a given pin in HR mode. The HR structure takes the information from the first instruction and ignores the remaining instructions.

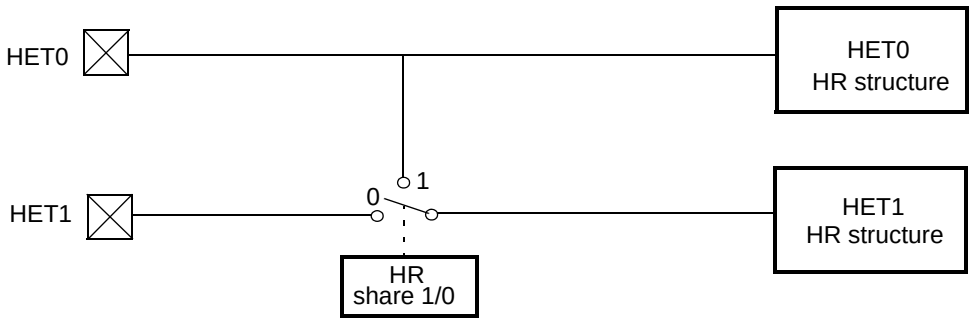
Figure 12-15. HR I/O Architecture



12.3.5.4 HR Structures Sharing

The HR share control register bits (see Section 12.6.10) allow 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 12-16.

Figure 12-16. Example of HR Structure Sharing for HET Pins 0/1



The following program gives an example how the HR share feature(HET0 HR structure and HET [1] HR structure shared) can be used for the PCNT instruction:

```
L00 PCNT { next=L01, type=rise2fall, pin=CC0 }
L01 PCNT { next=L00, type=fall2rise, pin=CC1 }
```

The HET [1] HR structure is also connected to the HET [0] pin. The L00_PCNT data field captures a high pulse and the L01_PCNT captures a low pulse on the **same** pin (HET [0] pin).

12.3.5.5 XOR-Shared HR Structure

Usually the HET design allows only one HR structure to cause HR edges on a pin configured as output pin. According to Section 12.3.5.10,the duration between two HR edges (both of which do not line up with the LR clock) cannot be smaller than 1 LRP. To eliminate this constraint, the HETXOR register allows a logical XOR of the output signals of two consecutive HR structures; see Figure 12-17. In this way, it is possible to generate pulses smaller than the loop resolution clock since both edges are now based on the HR clock. This is especially useful for a symmetrical PWM. See Figure 12-18.

This apparatus consists of 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 12-17. XOR-shared HR I/O

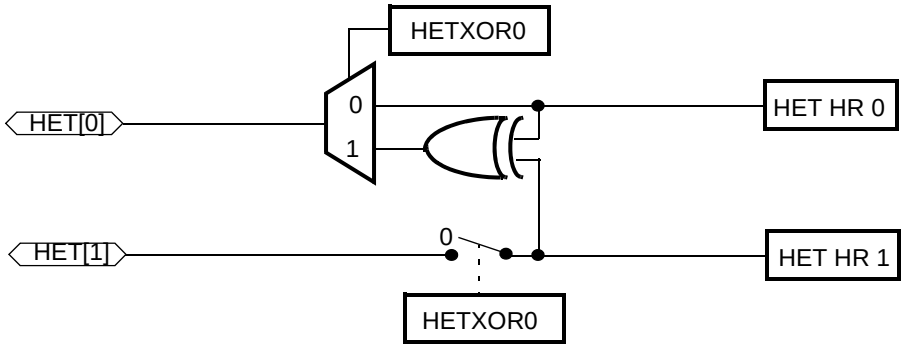
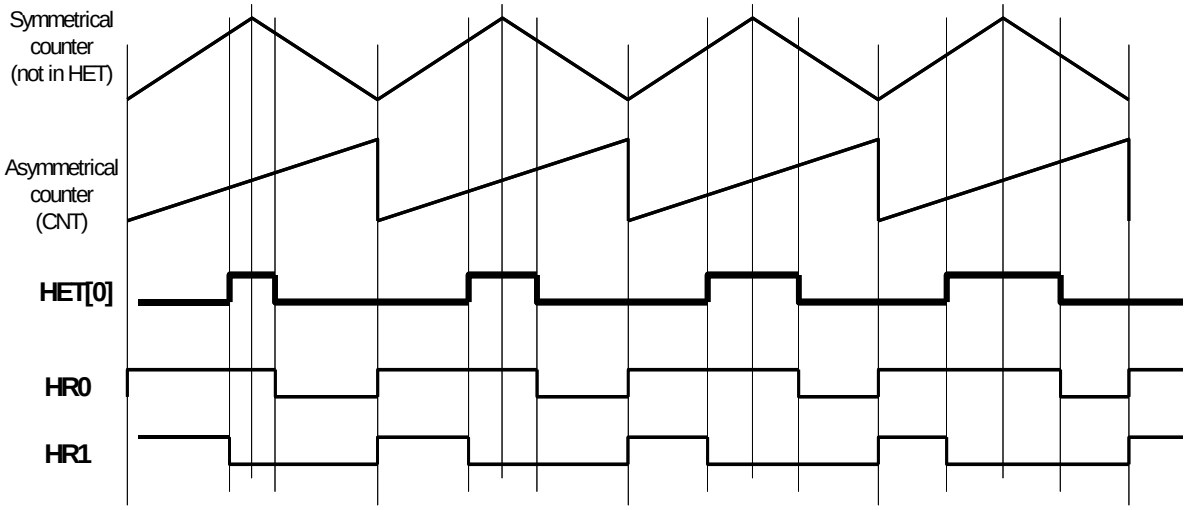


Figure 12-18. Symmetrical PWM with XOR-sharing Output



The following HET program gives an example for **one** channel of the symmetrical PWM. The generated timing is given in Figure 12-18.

```
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=19 }
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 }
```

The HET settings and output signal calculation for this example program are as follows:

- Pin HET[0] and HET[1] are XOR-shared.
- HETPFR[31:0] register = 0x501 → Lr = 32 and Hr = 2 → time slots Ts = 64
- PWM period (determined by CNT_max field) = (22+1) * LRP = 736 HRP
- Length of high pulse of (HET[0] XOR HET[1]) =

  LH = (17*LRP+19*HRP)-(5*LRP+13*HRP)

  With LR = 32, there is LRP = 32 * HRP, therefore,

  LH = (563-173) * HRP = 390 HRP
- Duty cycle (DC) = LH / PWM_period = 390 HRP / (736*HRP) = 53.0%

Figure 12-18 shows the implementation of the XOR-shared feature. The first two waveforms (symmetrical counter and CNT) show a symmetrical counter and an asymmetrical counter. The symmetrical counter is shown only to highlight the axis of symmetry and is not implemented in the HET. The asymmetrical counter, which is implemented with a CNT instruction, needs to be set to the period of the symmetrical 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.

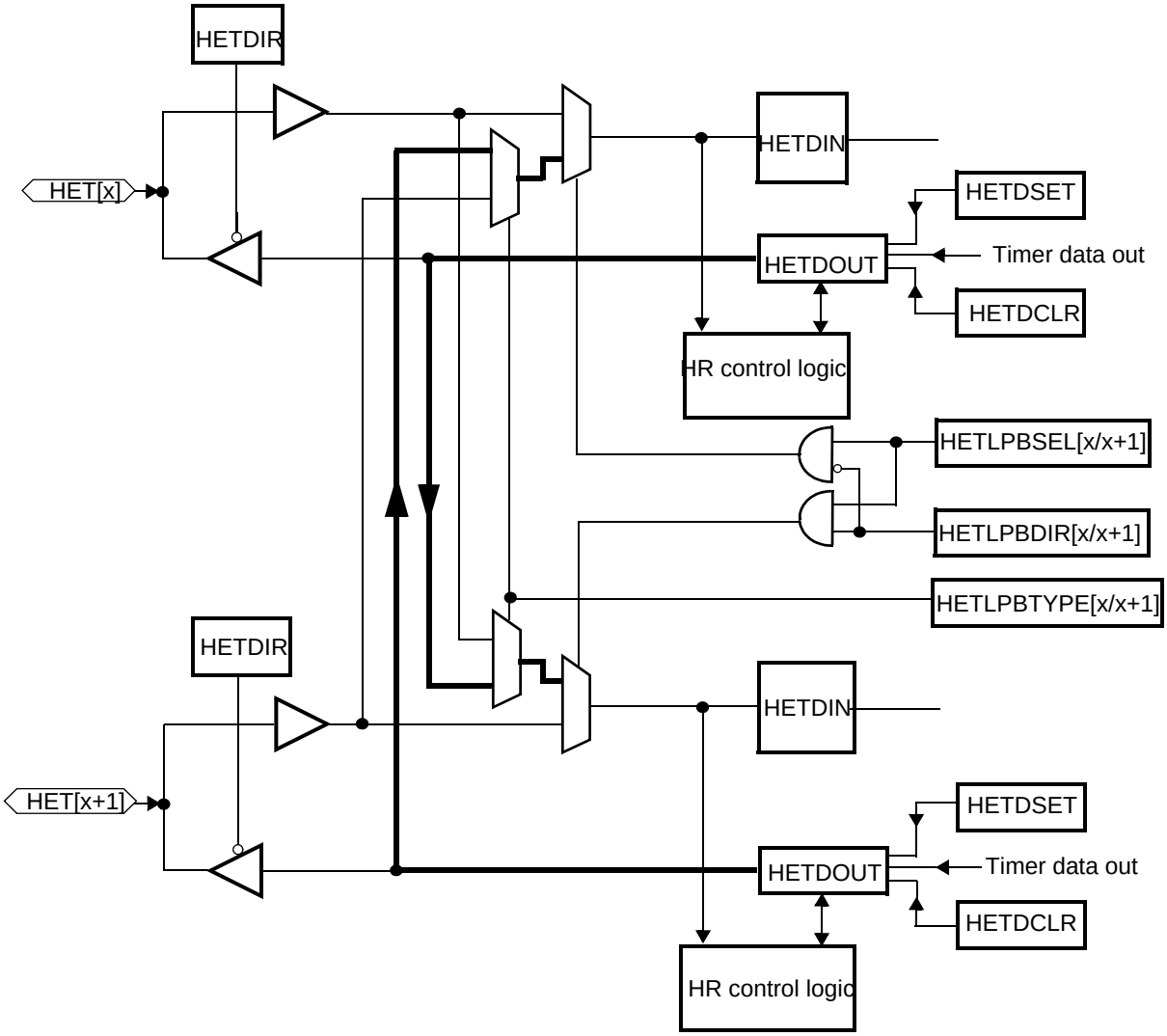
12.3.5.6 Loopback Modes

It is possible to output a signal on one pin (e.g., with ECMP; Section 12.7.3.10) and measure back the signal on the same pin (e.g., with ECMP). However, if HR is used to output the signal, it is not possible to measure

back the signal with HR, since the same HR logic cannot be used for both signal generation and signal measurement at the same time. It is only possible to measure back with LR.

The HET, however, provides two different loopback modes (digital loopback and analog loopback) to allow the signal to also be measured in HR.

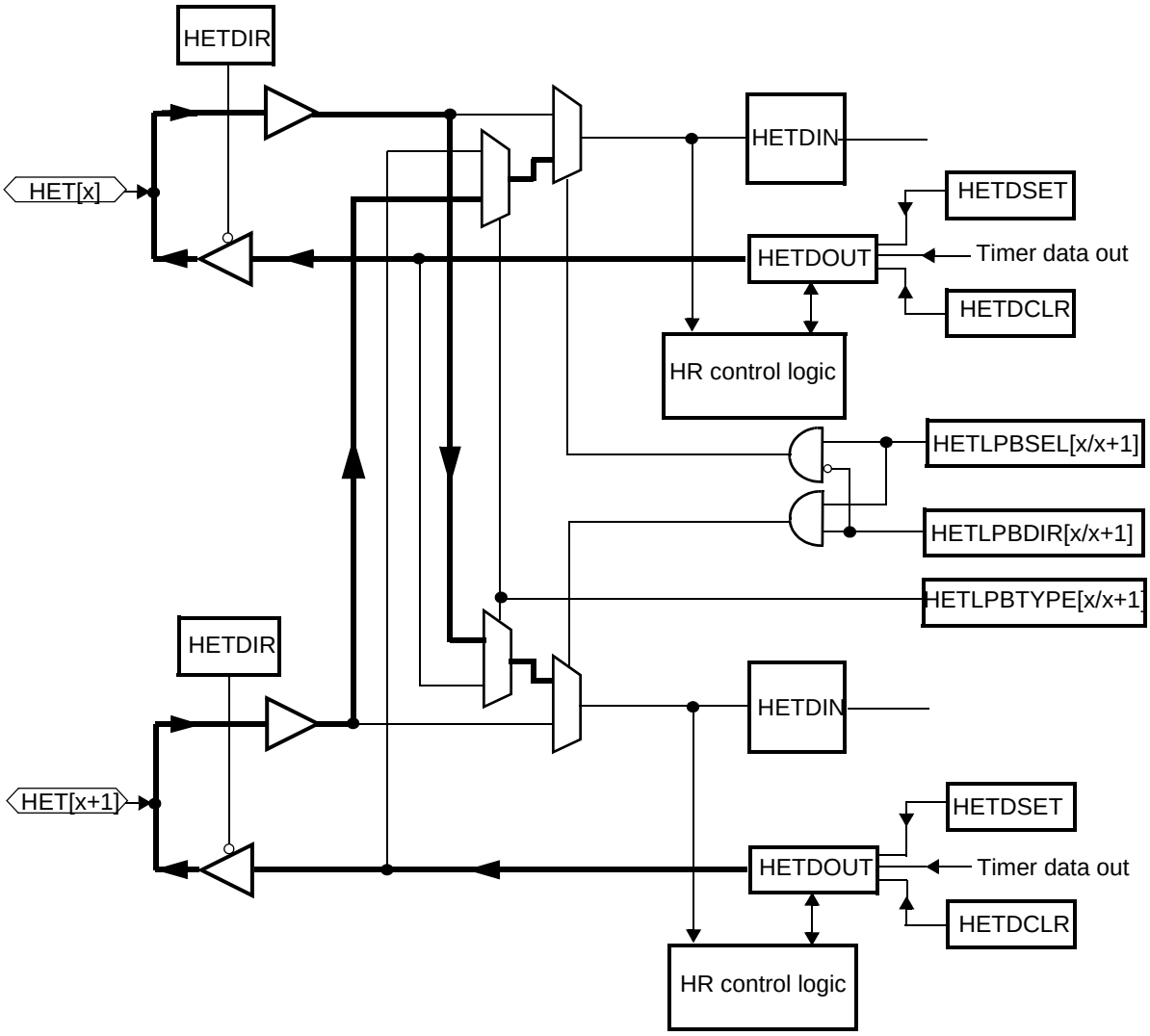
Figure 12-19. Digital Loopback I/O



To select digital loopback mode, the corresponding HETLPBSEL bit (Section 12.6.20) has to be set to 1 and HETLPBTYPE (Section 12.6.20) has to be set to 0. Depending on the setting of the HETLPBDIR bit (Section 12.6.21), the direction will go from HET[x] to HET[x+1] or from HET[x+1] to HET[x]. In digital loopback mode, the input of the output buffer will be connected to the input of HR logic and the HETDIN register.

Depending on the HETDIR bit setting, the signal to be fed back can be output on the pin or suppressed. The pin structure that is used to output the signal can still be used as general-purpose input if the HETDIR bit is set so that input mode is selected. The pin structure that measures back the generated signal can still use its pin as general-purpose output pin.

Figure 12-20. Analog Loopback I/O



To select analog loopback mode, the corresponding HETLPBSEL bit (Section 12.6.20) has to be set to 1 and HETLPBTYPE (Section 12.6.20) has to be set to 1. Depending on the setting of the HETLPBDIR bit (Section 12.6.21), the direction will go from HET[x] to HET[x+1] or from HET[x+1] to HET[x]. In analog loopback mode, the output of the input buffer will be connected to the input of HR logic and the HETDIN register.

The HETDIR bit has to be set to 1 so that the signal to be fed back can be output on the pin. The pin structure that measures back the generated signal can still use its pin as a general-purpose output pin.

12.3.5.7 HR/Low Resolution Bit

Four HR instructions—ECMP (Section 12.7.3.10), MCMP (Section 12.7.3.12), PWCNT (Section 12.7.3.16), and WCAP (Section 12.3.5.13)—have a dedicated hr_lr bit (HR/LR) allowing operation either in HR mode or in LR mode by ignoring the HR field. The hr_lr bit is P(7) (program field bit 7). By default, the hr_lr bit value is 0, which implies HR operation mode. However, setting this bit to 1 allows the use of several HR instructions on a single HR pin. Only one instruction will be allowed to operate in HR mode (i.e., the bit set to 0), but the other instructions can be used in LR mode (i.e., the bit set to 1).

12.3.5.8 ECMP Execution Example (in HR Mode)

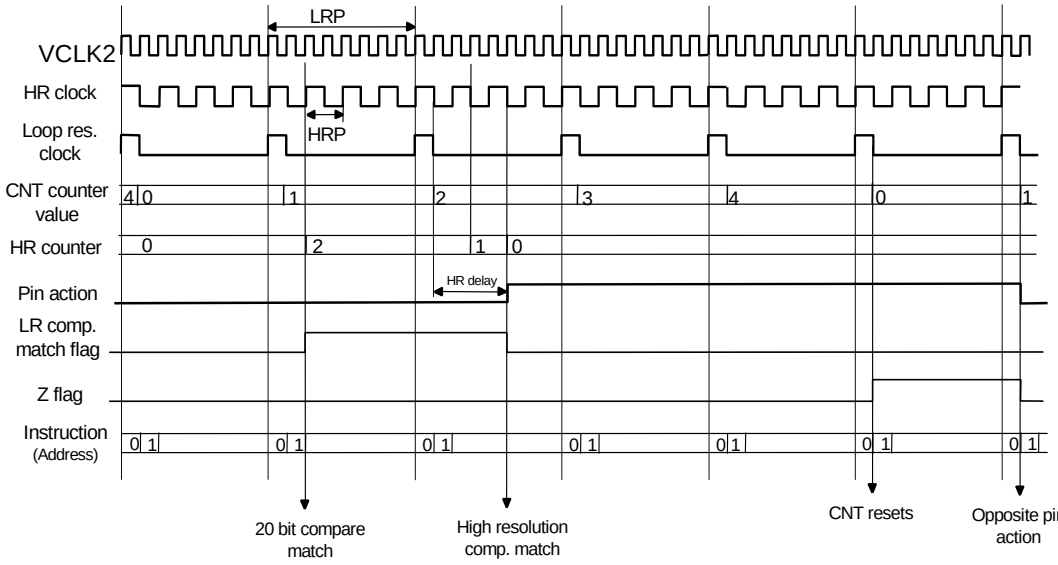
First is an example of an HR ECMP with a linear prescale of /2 and a loop resolution prescale of /4. With a 32-MHz VCLK2 clock, these parameters result in a 62.5 ns HR clock and a 0.25us resolution loop clock. This allows eight time slots for the program, which in this case will consist of one CNT and one ECMP instruction

(Section 12.7.3.7 and Section 12.7.3.10). The ECMP instruction is loaded into the HET RAM with a 25-bit compare value, the lower 5 bits representing the HR compare value.

When the 20-bit (low resolution) compare matches, the LR comp match flag will be set in the HR hardware for the corresponding pin and the HR compare value will be loaded from the five lower bits of the instruction data field to the HR counter. At the next LR clock, the HR counter will count down at the HR clock frequency and perform the set action when it reaches zero.

Figure 12-21 shows the execution timing when the 20-bit compare value is 1 and the HR compare value is 2.

Figure 12-21. ECMP Execution Timings



The following HET program gives an example for a one channel PWM. The generated timing is given in Figure 12-21.

```
HETPFR[31:0] register = 0x201 → lr=4 and hr=2 → time slots ts = 7
L00   CNT  {next=01h, reg=A, irq=OFF, max = 4 }
L01   ECMP {next=00h, en_pin_action=ON, cond_addr= 00h, pin=CC0, action=PULSEHI, reg=A,
irq=OFF, data= 1, hr_data = 0x10 }
; 20 bit compare value is 1 and the HR compare value is 2
```

Note: ECMP Opposite Actions

ECMP opposite actions are always synchronized to the loop resolution clock.

Capture/compare pins can be selected individually with instructions ECMP (Section 12.7.3.10), SCMP (Section 12.7.3.18), MCMP (Section 12.7.3.12), PWCNT (Section 12.7.3.16), MOV64 (Section 12.7.3.14), DADM64 (Section 12.7.3.8), and RADM64 (Section 12.7.3.17). A pin already defined as an output compare can be selected as an internal input in the instructions BR (Section 12.7.3.6), CNT (Section 12.7.3.7), SHFT (Section 12.7.3.20), and WCAP (Section 12.7.3.21).

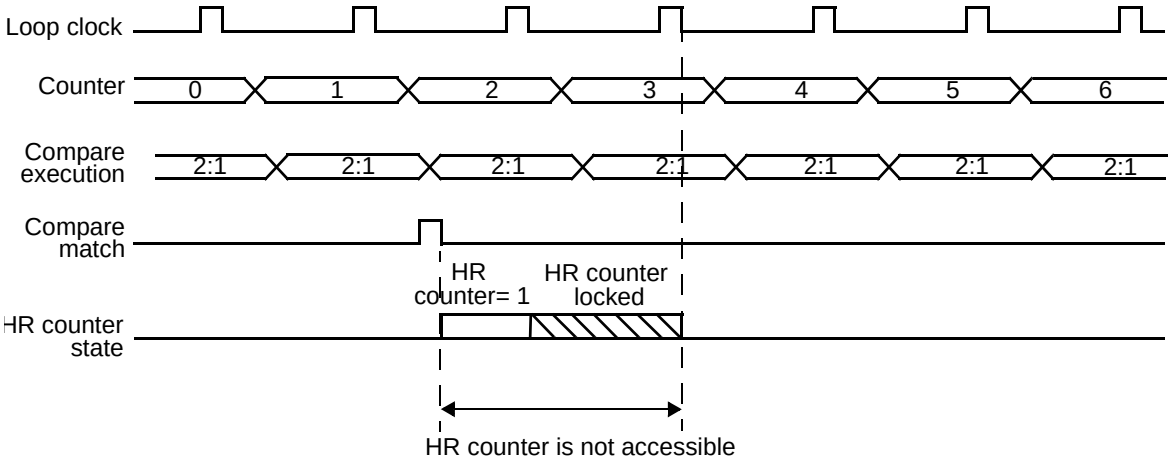
12.3.5.9 MCMP Execution Example

The MCMP instruction (see Section 12.7.3.12 for detailed information) can also be used in HR mode. Operation with the MCMP instruction is exactly the same as for the ECMP instruction except that the 20-bit low resolution is now the result of a magnitude compare rather than of an equality compare. Otherwise, operation is the same as for the 20-bit match setting the HR flag and loading the HR counter. The HR counter starts to decrement after the next resolution clock and performs the specified pin action. See Figure 12-22.

12.3.5.10 Limitation on ECMP and MCMP Operations in HR Mode

There is some limitation on the ECMP (Section 12.7.3.21) and MCMP (Section 12.7.3.12) operations in the HR mode, which Figure 12-22 illustrates.

Figure 12-22. ECMP Limitation Timing Diagram



ECMP or MCMP compares its own data field value with a register value that is loaded by a CNT instruction. The compare is a 20-bit comparison, even if ECMP or MCMP has an HR value. If this 20-bit comparison matches and if the HR mode is used, the HR compare value is loaded in the dedicated HR counter. The pin action happens in the next loop resolution cycle. The HR counter is locked during all this time, as shown in Figure 12-22.

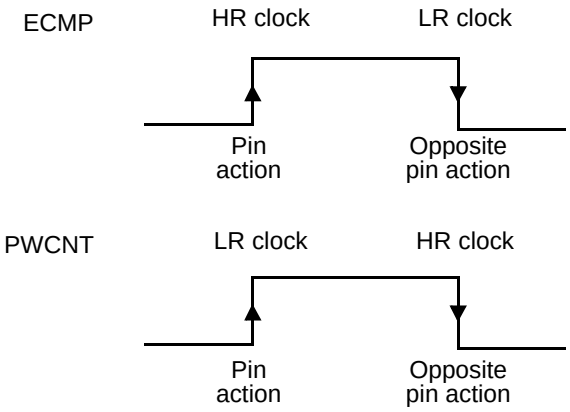
In other words, when a compare matches, one loop resolution cycles must pass before a new compare (in HR mode) can occur.

Note:
If it is necessary to have a more accurate pulse time, using the HR XOR-shared features with two HR pins will work around this limitation.

12.3.5.11 PWCNT Execution Example (in HR Mode)

The PWCNT instruction (Section 12.7.3.16) may also be used in HR mode to generate pulse outputs with HR width. The PWCNT instruction operates conversely to the ECMP instruction; see Figure 12-23. 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 12-23. High/Low Resolution Modes for ECMP and PWCNT



12.3.5.12 PCNT Execution Example (in HR Mode)

In conjunction with HR I/O pins, the PCNT instruction (Section 12.7.3.15) can capture an HR measurement of the high/low pulse time or periods of the input. As shown in Figure 12-24, at the first marker, the input goes

high and the HR counter immediately begins to count. The counter increments and rolls over until the falling edge, where it captures the counter value into the HR capture register (the second marker). The PCNT instruction begins counting when the synchronized input signal goes high and captures both the 20-bit data field and the HR capture register into RAM when the synchronized input falls (third marker).

Note:
The HR capture value written into RAM is shifted appropriately depending on the loop resolution prescale divide rate (lr).

Figure 12-24 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 12-24. PCNT Instruction Timing (With Capture Edge After HR Counter Overflow)

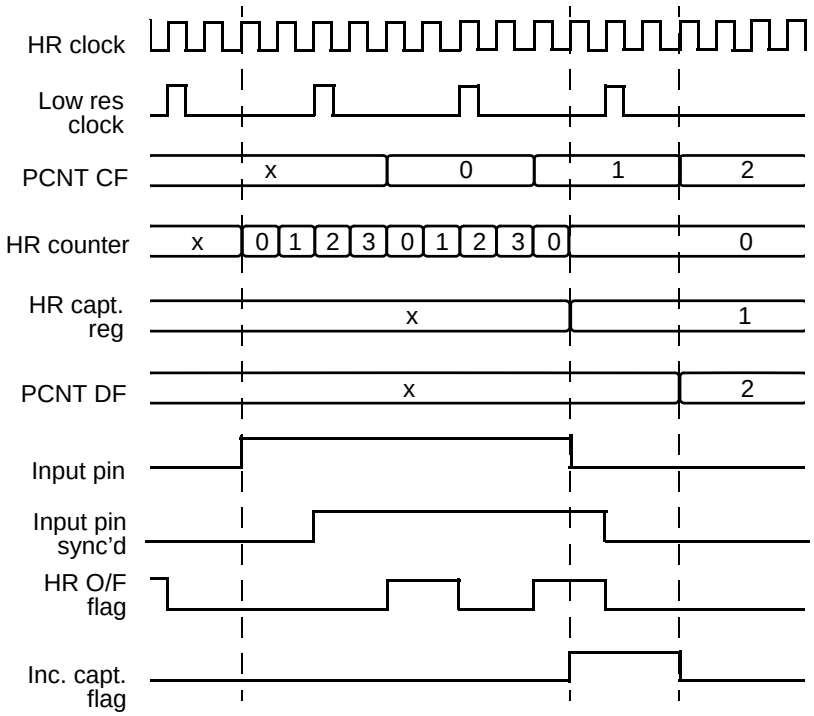
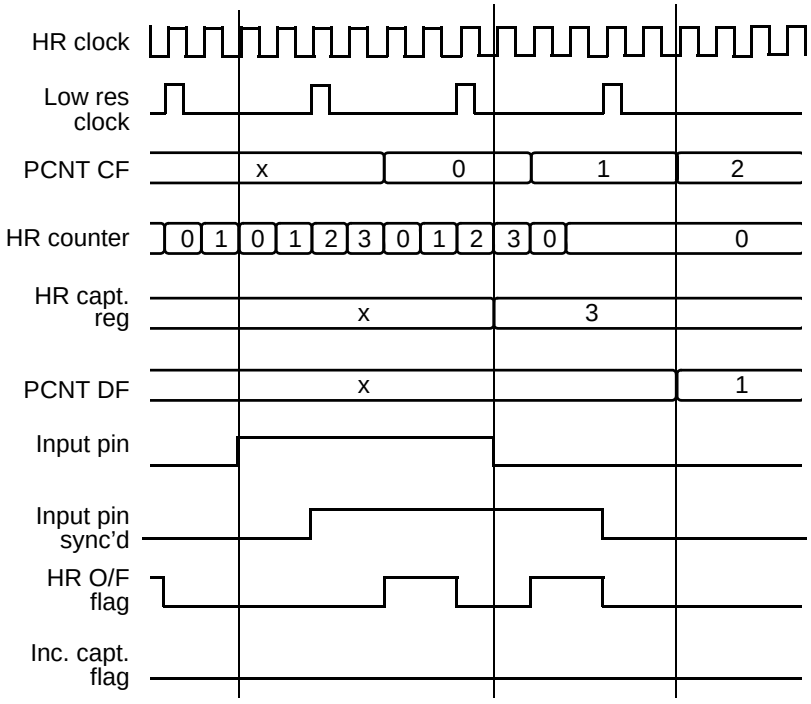


Figure 12-25 shows what happens when the capture edge arrives *before* the HR counter overflows. This causes the non-incremented value to be captured by the PCNT instruction.

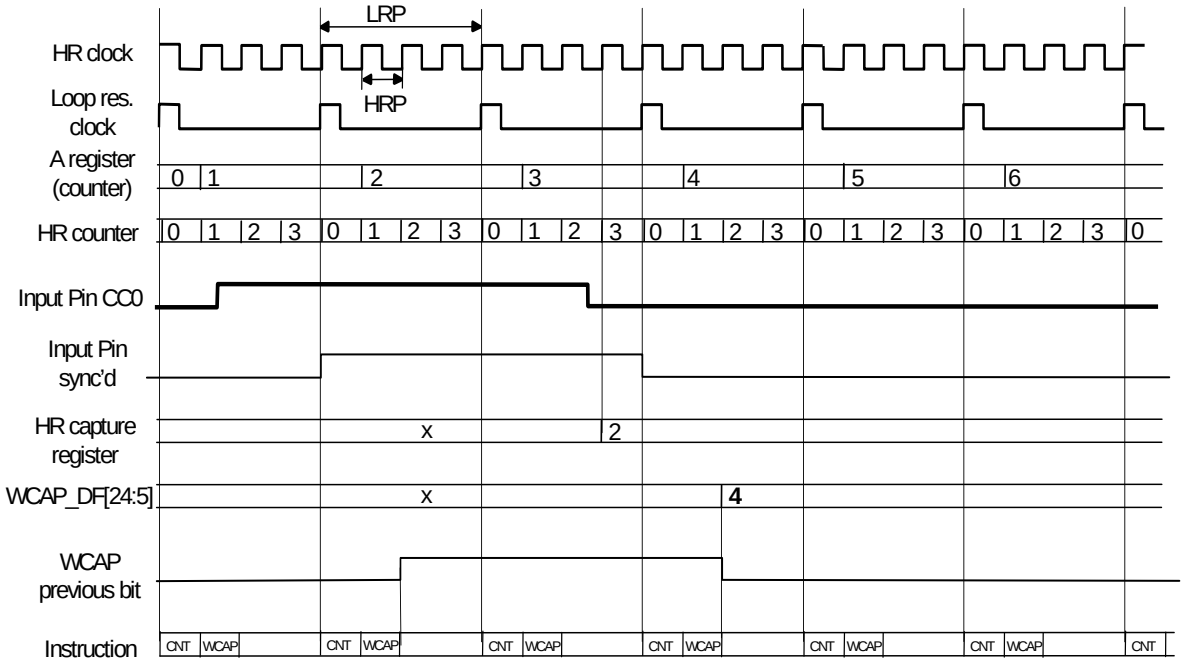
Figure 12-25. PCNT Instruction Timing (With Capture Edge Before HR Counter Overflow)



12.3.5.13 WCAP Execution Example (in HR Mode)

The WCAP instruction (Section 12.7.3.21) can be implemented for HR I/O. In this case the HR counter is always enabled (using the count always flag in the HR structure) 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 a free run timer saved in a register (for example, register A shown in Figure 12-26).

Figure 12-26. WCAP Instruction Timing



The following HET program gives an example for a one channel capture input using the WCAP instruction. The generated timing is given in Figure 12-26.

```
HETPFR_register = 0x0200 → lr = 4, hr = 1, ts = 3

L00 CNT  {reg=A, max=0fffffh}
L01 WCAP {next=L00, cond_addr=L00, hr_lr=high, reg=A, event= FALL, pin=CC0,
          data=0}
```

In the timing of the example, the WCAP is configured to capture the counter when a **falling** edge occurs. The WCAP data field (WCAP_DF) is complete in the loop succeeding the loop, in which the edge occurred. In the figure example, the current value of the counter (4) is captured to WCAP_DF[24:5] and the value of the HR capture register (2) is transferred to the valid bits (according the lr prescaler) of WCAP_DF[4:0]. Therefore, in the example 0x090 is captured to WCAP_DF[24:0].

12.3.6 Interrupts and Exceptions

HET interrupts are 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 HETFLG register. The address code for this flag is determined by the five LSBs of the current timer program address.

Note:
Enabling or disabling the interrupt generation of certain instructions (BR, ECNT, SHFT, WCAP) when the HET is turned on could falsify the operation of these instructions. This effect is due to the possible modification of the PRV bit by the HET during the read/modify/write of the CPU to modify the control field.

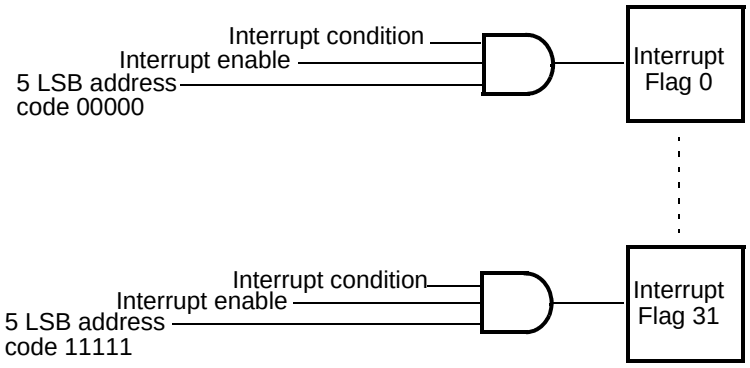
Table 12-5. Interrupt Sources and Corresponding Offset Values in Registers HETOFFx

Source No.	Offset Value
no interrupt	0
Instruction 0, 32, 64, 96, 128, ..., 224	1
Instruction 1, 33, 65, 97, 129,..., 225	2
:	:
Instruction 31, 63, 95, 127,..., 255	32
Program overflow	33
APCNT underflow:	34
APCNT overflow	35

To execute the interrupt service routine, the main CPU must first determine which source created the interrupt request. It does that by reading the offset register (HETOFFx), which gives the number of the source. Reading the offset register will automatically clear the source flag that created the request. However, if it is decided not to use the offset register to check the interrupt source, the flag should still be cleared after the interrupt has been serviced.

The instructions capable of generating interrupts are listed in Table 12-38 in Section 12.7.

Figure 12-27. Interrupt Servicing



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. Two interrupt requests are generated by the HET for the vectored interrupt manager module (VIM). Figure 12-27 shows how the interrupts are serviced.

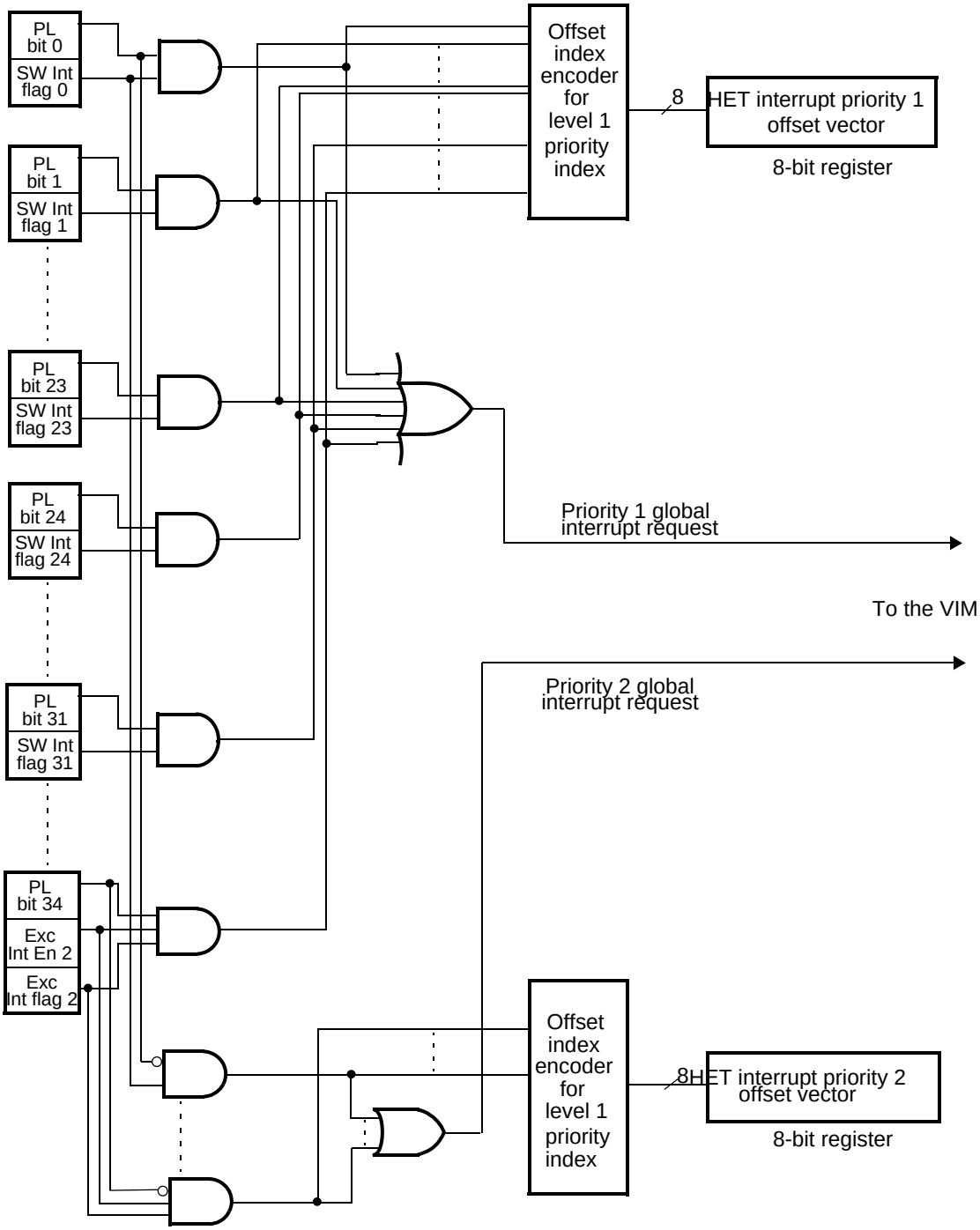
The HET can generate three types of exception from the following:

- Program overflow
- APCNT underflow (see Section 12.5.1.2)
- APCNT overflow (see Section 12.5.1.3)

12.3.7 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 lowest offset value. This scheme is hard-wired in the offset encoder. See Figure 12-28.

Figure 12-28. Interrupt Flag/Priority Level Architecture



12.4 HET Parity

The HET RAM includes an optional parity feature. This feature is implemented as a single-bit. Whether to check for even or odd parity can be configured in the system module. The parity bit is mapped at an offset of 1000h from the HET RAM base address. For every HET RAM word, a parity bit is calculated and stored into the least significant bit (LSb) of a 32-bit word in the parity region located at an offset of 1000h from that HET RAM word.

Table 12-6 describes the HET parity region address map.

Table 12-6. Parity Region Address Map

Instruction Address	Program Field Parity Address	Control Field Parity Address	Data Field Parity Address	Reserved Address
00h	X1000h	X1004h	X1008h	X100Ch
01h	X1010h	X1014h	X1018h	X101Ch
02h	X1020h	X1024h	X1028h	X102Ch
:	:	:	:	:
:	:	:	:	:
7Fh	X17F0h	X17F4h	X17F8h	X17FCh

Note: RAM
HET Parity region is memory mapped from 1000h to 17FFh.

In order to test the parity generation/comparison logic, a mechanism has been provided to corrupt the parity bits in HET RAM. Based on the TEST bit in the HETPCR register, the CPU is able to access/modify the parity bits in HET RAM. When in test mode, no parity checking will be done while reading from parity memory, however, parity checking will be performed while reading from normal HET RAM. A normal write to the HET RAM will also write to the associated parity bits. Since the HET supports only word writes (21, 22 or 25 bits all together) and never bit/byte or half word writes, read-modify-write need not be implemented for storing parity information.

If the TEST bit in the HETPCR register is set, then the Parity Address register holds the address of the first parity error generated in the HET RAM. This register is frozen from being updated until the CPU has read the value. If more than one HET RAM location (Control, Data & Program) shows a parity error in the same cycle, then only the lower word address will be captured in the Parity Address Register.

12.5 Angle Functions

Engine management systems require an angle-referenced time base to synchronize signals to the engine toothed wheel. The HET has a method to provide such a time base:

- A software angle counter for low-end engine systems. The reference is created by the HET using three dedicated instructions with fractional angle steps equal to /8, /16, /32, /64.

12.5.1 Software Angle Generator (SWAG)

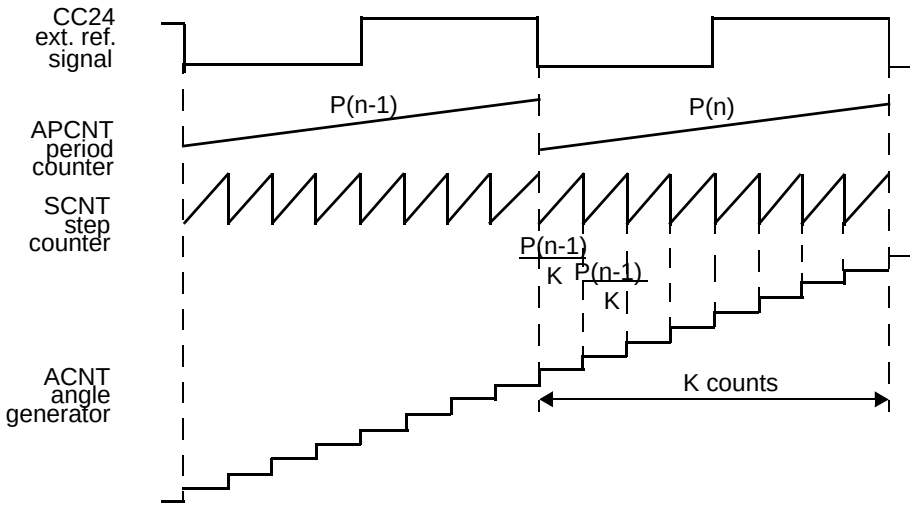
The HET provides three specialized count instructions to generate an angle-referenced time base synchronized to an external reference signal (the toothed wheel signal) that defines angular reference points.

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 you 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 (Section 12.7.3.5) incremented on each resolution clock measures the periods P(n) of the external signal. The second counter SCNT (Section 12.7.3.19) 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 (Section 12.7.3.2) accumulates the fractions generated by SCNT.

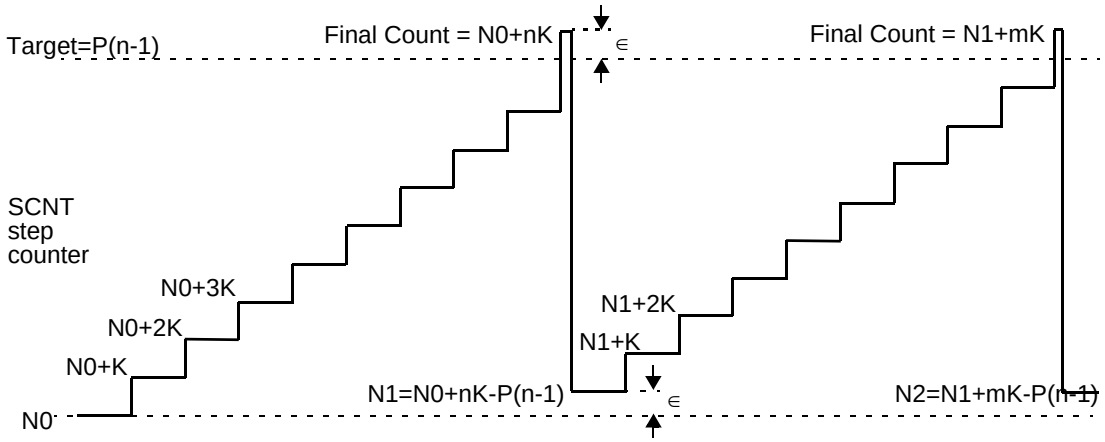
A HET timer program can only have one angle generator. Figure 12-29 illustrates the basic operation of APCNT, SCNT, and ACNT.

Figure 12-29. Operation of HET Count Instructions



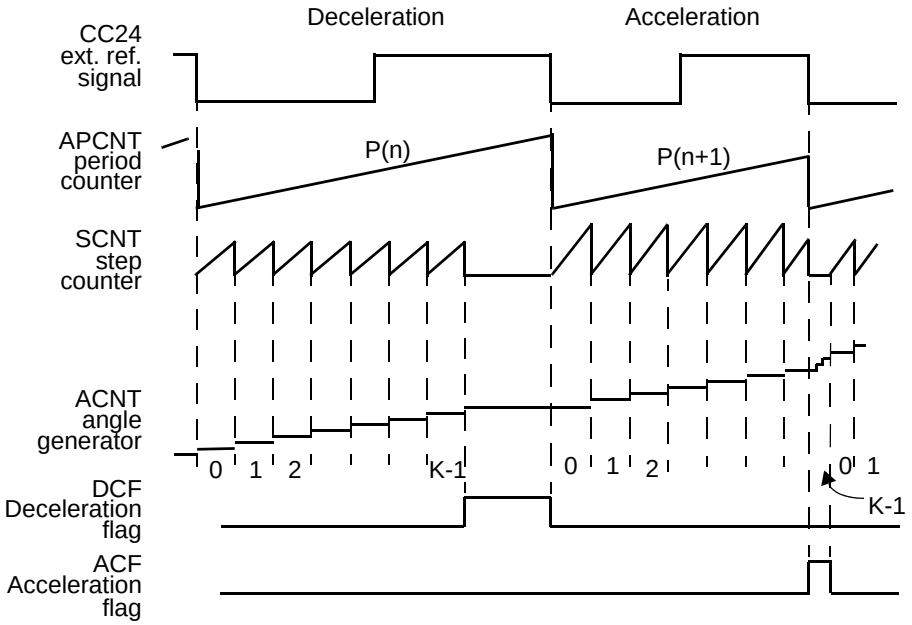
Because of stepping, the final count of SCNT does not usually exactly match the target value P(n-1). Figure 12-30 illustrates how SCNT compensates for this feature by starting each cycle with the remainder (final count - target) of the previous cycle.

Figure 12-30. 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 12-31 illustrates how the compensations for acceleration and deceleration operate.

Figure 12-31. ACNT Period Variation Compensations



12.5.1.1 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 (Section 12.7.3.19) and ACNT (Section 12.7.3.2). 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 12-32 and Figure 12-33 illustrate the behavior of the angle generator during a gap after a deceleration or acceleration of the HET.

Figure 12-32. HET Timings Associated with the Gap Flag (ACNT Deceleration)

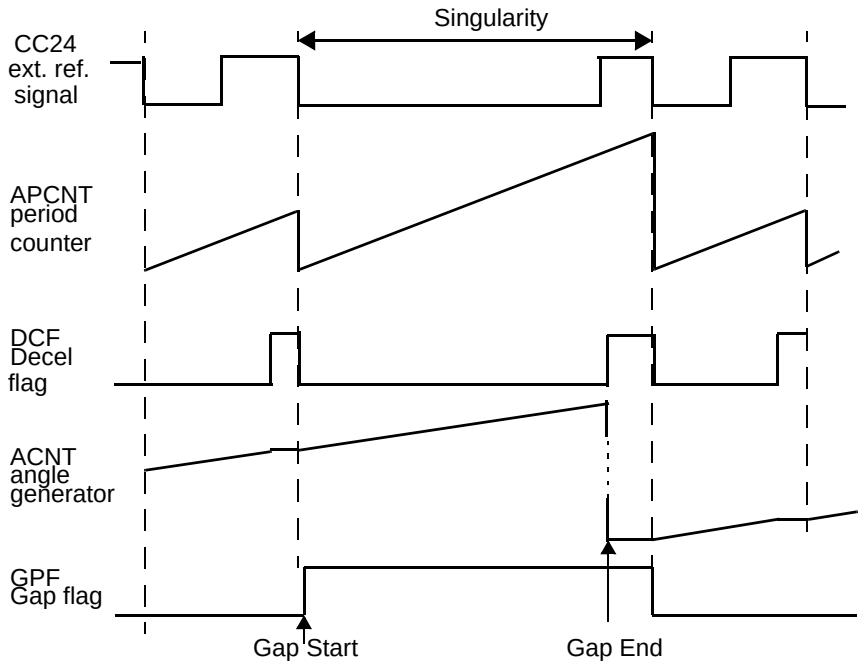
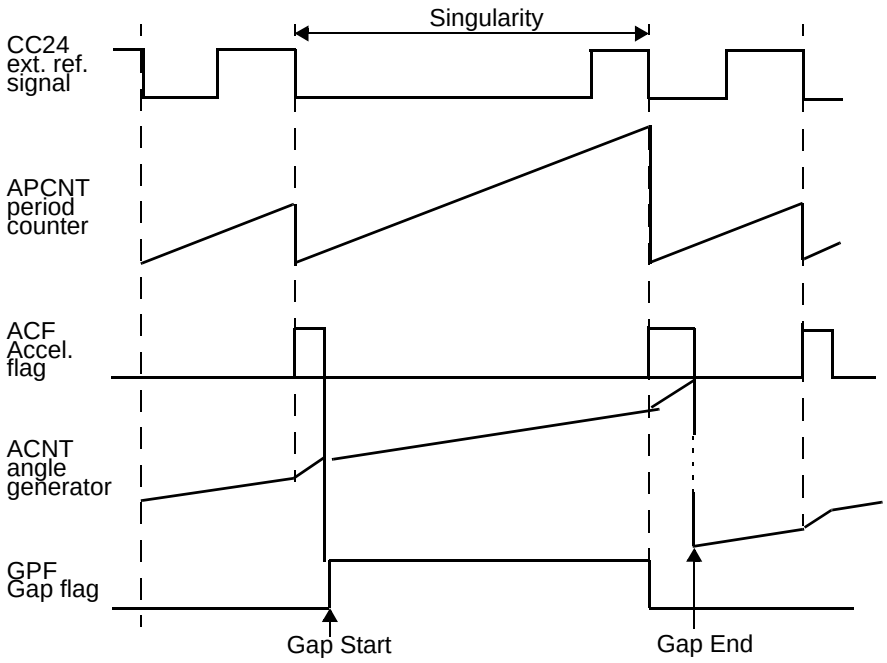


Figure 12-33. HET Timings Associated with the Gap Flag (ACNT Acceleration)



12.5.1.2 APCNT Underflow

The fastest valid external signal APCNT (Section 12.7.3.5) can accept must satisfy the following condition:

$$\text{Step Width K} < \frac{\text{Period Min.}}{\text{Resolution (LRP)}}$$

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.

12.5.1.3 APCNT Overflow

The slowest valid external signal APCNT (Section 12.7.3.5) can measure must satisfy the following condition:

$$\frac{\text{Period Max}}{\text{Resolution}} < 1048575$$

When this limit is reached (APCNT count equals all 1s), 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.

12.6 HET Control Registers

Accesses to the peripheral registers can be done in byte, half-word, or word format.

For the registers having a read/clear, read/set structure, the update is as follows:

- Writing a 1 performs the action (set or clear depending on the structure).
- Writing a 0 has no effect (no change for the bit).

Table 12-7. Read/Write Conventions

Action	Description
R	Read
W	Write
C	Clear
S	Set
n	Value after reset
RWP	Read in all modes, write in privileged mode only

Figure 12-34 presents a summary of the HET registers.

The base address for the control registers is 0xFFF7 B800

Figure 12-34. HET Register Summary

Offset Address Register(1)(2)	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
0x00 HETGCR Page 628	Reserved							Power Down	Reserved							CLK Master/ Slave
	Reserved							64-Bit Access	Reserved					Debug Status Flag	Ignore Susp.	Turn On/Off
0x04 HETPFR Page 630	Reserved															
	Reserved					LRPres.Factor(2–0)			Reserved			HRPres.Factor(5–0)				
0x08 HETADDR Page 632	Reserved															
	Reserved								HETADDR(7–0)							
0x0C HETOFF1 Page 633	Reserved															
	Reserved								OFFSET1(7–0)							

1 The physical address (base + offset) of these registers is device specific. See the device-specific data sheet to verify the register base address.

2 The HETXOR register is only available for certain TMS470M Series derivatives. Refer to the device-specific data sheet.

Figure 12-34. HET Register Summary

Offset Address Register[1][2]	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
0x10 HETOFF2 Page 634	Reserved															
	Reserved								OFFSET2(7–0)							
0x14 HETEXC1 Page 635	Reserved							APCNT OVRFL ENA	Reserved							APCNT UNRFL ENA
	Reserved							PRGM OVRFL ENA	Reserved					APCNT OVRFL PRY	APCNT UNRFL PRY	PRGM OVRFL PRY
0x18 HETEXC2 Page 636	Reserved															
														APCNT OVRFL FLAG	APCNT UNRFL FLAG	PRGM OVRFL FLAG
0x1C HETPRY Page 637	HETPRY[31:16]															
	HETPRY[15:0]															
0x20 HETFLG Page 638	HETFLAG[31:16]															
	HETFLAG[15:0]															
0x2C HETHRSH Page 639	Reserved															
	Reserved							HR SHARE 23/22	HR SHARE 21/20	HR SHARE 19/18	HR SHARE 17/16	HR SHARE 15/14	HR SHARE 13/12	HR SHARE 11/10	HR SHARE 9/8	HR SHARE 7/6	HR SHARE 5/4	HR SHARE 3/2	HR SHARE 1/0
0x30 HETXOR Page 640	Reserved															
	Reserved							XOR SHARE 23/22	XOR SHARE 21/20	XOR SHARE 19/18	XOR SHARE 17/16	XOR SHARE 15/14	XOR SHARE 13/12	XOR SHARE 11/10	XOR SHARE 9/8	XOR SHARE 7/6	XOR SHARE 5/4	XOR SHARE 3/2	XOR SHARE 1/0
0x34 HETDIR Page 641	HETDIR[31:16]															
	HETDIR[15:0]															

1 The physical address (base + offset) of these registers is device specific. See the device-specific data sheet to verify the register base address.
2 The HETXOR register is only available for certain TMS470M Series derivatives. Refer to the device-specific data sheet.

Figure 12-34. HET Register Summary

Offset Address Register[1][2]	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
0x38 HETDIN Page 642	HETDIN[31:16]															
	HETDIN[15:0]															
0x3C HETDOUT Page 643	HETDOUT[31:16]															
	HETDOUT[15:0]															
0x40 HETDSET Page 644	HETDSET[31:16]															
	HETDSET[15:0]															
0x44 HETDCLR Page 645	HETDCLR[31:16]															
	HETDCLR[15:0]															
0x48 HETPDR Page 646	HETPDR[31:16]															
	HETPDR.15:0															
0x4C HETPULDIS Page 647	HETPULDIS[31:16]															
	HETPULDIS[15:0]															
0x50 HETPSL Page 648	HETPSL[31:16]															
	HETPSL[15:0]															
0x60 HETLPBSEL Page 649	HETLP BTYPE 31/30	HETLP BTYPE 29/28	HETLP BTYPE 27/26	HETLP BTYPE 25/24	HETLP BTYPE 23/22	HETLP BTYPE 21/20	HETLP BTYPE 19/18	HETLP BTYPE 17/16	HETLP BTYPE 15/14	HETLP BTYPE 13/12	HETLP BTYPE 11/10	HETLP BTYPE 9/8	HETLP BTYPE 7/6	HETLP BTYPE 5/4	HETLP BTYPE 3/2	HETLP BTYPE 1/0
	HETLP BSEL 31/30	HETLP BSEL 29/28	HETLP BSEL 27/26	HETLP BSEL 25/24	HETLP BSEL 23/22	HETLP BSEL 21/20	HETLP BSEL 19/18	HETLP BSEL 17/16	HETLP BSEL 15/14	HETLP BSEL 13/12	HETLP BSEL 11/10	HETLP BSEL 9/8	HETLP BSEL 7/6	HETLP BSEL 5/4	HETLP BSEL 3/2	HETLP BSEL 1/0

1 The physical address (base + offset) of these registers is device specific. See the device-specific data sheet to verify the register base address.
2 The HETXOR register is only available for certain TMS470M Series derivatives. Refer to the device-specific data sheet.

Figure 12-34. HET Register Summary

Offset Address Register[1][2]	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
0x64 HETLPBDIR Page 650	Reserved															
	HETLP BDIR 31/30	HETLP BDIR 29/28	HETLP BDIR 27/26	HETLP BDIR 25/24	HETLP BDIR 23/22	HETLP BDIR 21/20	HETLP BDIR 19/18	HETLP BDIR 17/16	HETLP BDIR 15/14	HETLP BDIR 13/12	HETLP BDIR 11/10	HETLP BDIR 9/8	HETLP BDIR 7/6	HETLP BDIR 5/4	HETLP BDIR 3/2	HETLP BDIR 1/0
0x68 HETPCR Page 651	Reserved															HET-STOP
	Reserved							TEST	Reserved				PARITY_ENA			
0x6C HETPIEN Page 653	Reserved															
	Reserved															INT EN
0x70 HETPIFLAG Page 654	Reserved															
	Reserved															INT FLAG
0x74 HETPAR Page 655	Reserved															
	Reserved					ERROR ADDRESS									0	0

1 The physical address (base + offset) of these registers is device specific. See the device-specific data sheet to verify the register base address.
2 The HETXOR register is only available for certain TMS470M Series derivatives. Refer to the device-specific data sheet.

12.6.1 Global Configuration Register (HETGCR)

Figure 12-35 and Table 12-8 illustrate the HETGCR register.

Figure 12-35. Global Configuration Register (HETGCR) [offset = 0x00]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved							Power Down	Reserved							clk_ master /slave
R-0							R/W-0	R-0							R/W-0
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							64-bit Access	Reserved					Debug Status Flag	Ignore Suspend	Turn On/Off
R-0							R/W-0	R-0					R/C-0	R/W-0	R/W-0

R = Read; W = Write; C = Clear; -*n* = value after reset

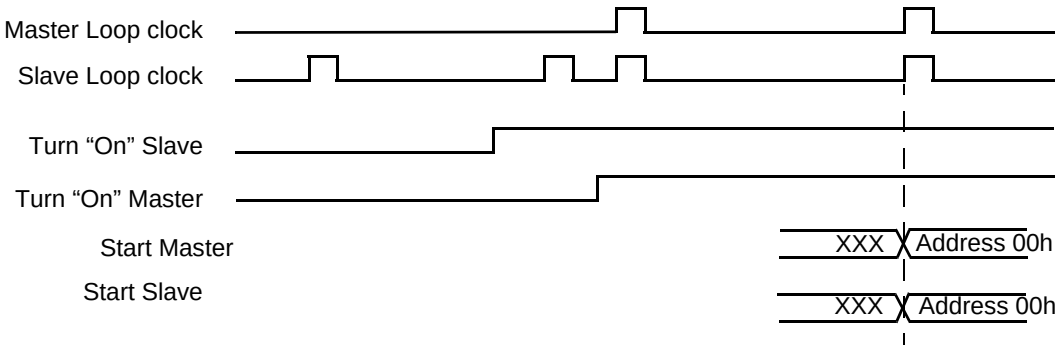
Table 12-8. Global Configuration Register (HETGCR) Field Descriptions

Bit	Name	Value	Description
31–25	Reserved		Reads return 0 and writes have no effect.
24	Power-Down		HET power-down mode bit. When Power-Down = 1, the HET internal clocks are stopped. This leaves the module in a static state in which it consumes the lowest possible current. To release the power-down mode, this bit must be written to 0; after setting turn-off, you must delay until the end of the timer program loop before putting the HET in power-down mode. This can be done by waiting until the HET PROGRAM ADDRESS becomes zero.
		0	HET clocks are running.
		1	HET clocks are stopped (power-down mode).
23–17	Reserved		Reads return 0 and writes have no effect.
16	clk_master/slave		This bit is used to synchronize multi-HETs. If set (HET is master), the HET outputs a signal to synchronize the prescalers of the slave HET. By default, this bit is reset, which means a slave configuration.
		0	HET is configured as a slave.
		1	HET is configured as a master.
			Note: This bit must be set to 1 for single-HET configuration.
15–9	Reserved		Reads return 0 and writes have no effect.
8	64-bit Access		Any operation mode (read/write):
		0	Timer RAM is accessed 32 bits at a time
		1	Timer RAM is accessed 64 bits at a time
7–3	Reserved		Reads return 0 and writes have no effect.
2	Debug Status Flag		This flag is set when the device test mode is set and a breakpoint is reached in the HET program. A debug request signal is asserted to the main CPU.
			Any operation mode (read):

Table 12-8. Global Configuration Register (HETGCR) Field Descriptions (Continued)

Bit	Name	Value	Description
		0	*Read:* No HET instruction with a breakpoint has been reached since the flag was last cleared. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* A HET instruction with a breakpoint has been reached since the flag was last cleared. *Write:* The bit is cleared.
1	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 HET continues operating.
		0	HET stops when a software breakpoint is encountered.
		1	HET ignores software breakpoints.
0	Turn On/Off		When turn on/off = 0, the timer program stops executing. Turn-off is automatically delayed until the current timer program loop is completed. After enabling Turn-off, you must delay until the end of the timer program loop before putting the HET in power-down mode. This can be done by waiting until the HET PROGRAM ADDRESS becomes zero.Turn-off does not affect the content of the timer RAM, ALU registers, or control registers. Turn-off resets all flags. See Figure 12-36.
		0	The HET is off. **Note: Turn-off 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 CPU reset, the timer is turned off by default.**
		1	The HET is on. **Note: When turn off/on equals 1, timer program execution starts synchronously to the Loop clock. In case of multiple HETs configuration, the slave HETs 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).**

Figure 12-36. Multiple HETs Turn-On Sequence



12.6.2 Prescale Factor Register (HETPFR)

Figure 12-37 and Table 12-12 present the HETPFR.

Figure 12-37. Prescale Factor Register (HETPFR) [offset = 0x04]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					LRPres.Factor(2–0)			Reserved		HRPres.Factor(5–0)					
R-0					R/W-0			R-0		R/W-0					

R = Read; WP = Write in any mode; *-n* = Value after reset

Table 12-9. Prescale Factor Register (HETPFR) Field Descriptions

Bit	Name	Value	Description
31–11	Reserved		Reads return 0 and writes have no effect.
10–8	LRPres.Factor(2–0)	0–7h	Loop resolution prescale factor code. The binary code programmed in the register gives the loop resolution (LR). The LR pre-scale factor code and the HR pre-scale factor code define the number of time slots available. See Table 12-10 for the loop resolution encoding format, calculated by: *Time Slots Available = HR Prescale Divide rate X Loop Resolution Prescale Divide rate*
8–6	Reserved		Reads return 0 and writes have no effect.
5–0	HRPres.Factor(5–0)	0–3Fh	HR prescale factor code. The binary code programmed in the register gives the HR. See Table 12-11 for the HR encoding format.

Table 12-10. Loop Resolution Encoding Format

Loop Resolution Prescale Factor Code			Loop-Res Prescale Divide Rate
2	1	0	
0	0	0	1
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	Reserved
1	1	1	Reserved

Table 12-11. HR Encoding Format

HR Prescale Factor code						HR Prescale Divide Rate
5	**4**	**3**	**2**	**1**	**0**	
0	0	0	0	0	0	1
0	0	0	0	0	1	2
0	0	0	0	1	0	3
0	0	0	0	1	1	4
0	0	0	1	0	0	5
:	:	:	:	:	:	:
:	:	:	:	:	:	:
1	1	1	1	0	1	62
1	1	1	1	1	0	63
1	1	1	1	1	1	64

12.6.3 HET Current Address Register (HETADDR)

Figure 12-38 and Table 12-12 present the register for HETADDR.

Figure 12-38. HET Current Address Register (HETADDR) [offset = 0x08]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								HETADDR(7–0)							
R-0								R-0							

R = Read; *-n* = Value after reset

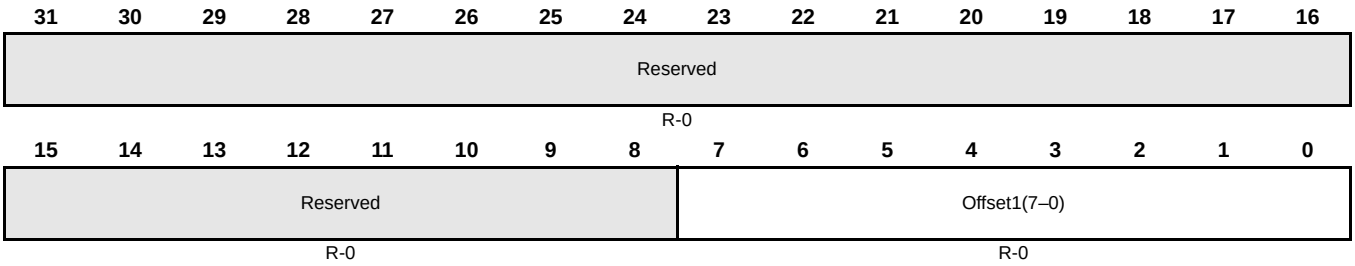
Table 12-12. HET Current Address Register (HETADDR) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return 0 and writes have no effect.
7–0	HETADDR(7–0)	0–FFh	HET address. A read of these bits in any operation mode returns the current HET program address register.

12.6.4 Offset Index Priority Level 1 Register (HETOFF1)

Figure 12-39 and Table 12-13 present the HETOFF1 register.

Figure 12-39. Offset Index Priority Level 1 Register (HETOFF1) [offset = 0x0C]



R = Read; -*n* = Value after reset

Table 12-13. Offset Index Priority Level 1 Register (HETOFF1) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return 0 and writes have no effect.
7–0	Offset(7–0)	0–FFH	Offset. HETOFF1[7:0] indexes the currently pending high-priority interrupt. Offset values and sources are listed below and the interrupt encoding format is presented in Table 12-14. *Any operation mode (read):* Read of these bits determines the pending HET interrupt; the corresponding flag (in the HETFLG) is also cleared. *Emulation mode (read):* Read of these bits determine the pending HET interrupt but the corresponding flag is not cleared.

Table 12-14. Interrupt Offset Encoding Format

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

12.6.5 Offset Index Priority Level 2 Register (HETOFF2)

Figure 12-40 and Table 12-15 describe the HETOFF2 register.

Figure 12-40. Offset Index Priority Level 2 Register (HETOFF2) [offset = 0x10]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								Offset2(7–0)							
R-0								R-0							

R = Read; *-n* = Value after reset

Table 12-15. Offset Index Priority Level 2 Register (HETOFF2) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return 0 and writes have no effect.
7–0	Offset2(7–0)	0–FFh	HETOFF2(7–0) indexes the currently pending low-priority interrupt. Offset values and sources are listed in Table 12-14. *Any operation mode (read):* A read of these bits in an operation mode determines the pending HET interrupt; the corresponding flag (in the HETFLG) is also cleared. *Emulation mode (read):* A read of these bits in emulation mode determines the pending HET interrupt, but the corresponding flag is not cleared.

12.6.6 Exception Control Register 1 (HETEXC1)

Figure 12-41 and Table describe the HETEXC1.

Figure 12-41. Exception Control Register (HETEXC1) [offset = 0x14]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved							APCNT Ovrfl Ena	Reserved							APCNT Undrfl Ena
R-0							R/W-0	R-0							R/W-0
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							Prgm Ovrfl Ena	Reserved					APCNT OVRfl Pry	APCNT Undrfl Pry	Prgm Ovrfl Pry
R-0							R/W-0	R-0					R/W-0	R/W-0	R/W-0

R = Read; W = Write; *-n* = Value after reset

Table 12-16. Exception Control Register (HETEXC1) Field Descriptions

Bit	Name	Value	Description
31–25	Reserved		Reads return 0 and writes have no effect.
24	APCNT Ovrfl Ena		APCNT overflow enable.
		0	The APCNT overflow exception is not enabled.
		1	The APCNT overflow exception is enabled.
23–17	Reserved		Reads return 0 and writes have no effect.
16	APCNT Undrfl Ena		APCNT underflow enable. Any operation mode (read/write):
		0	APCNT underflow exception is not enabled.
		1	APCNT underflow exception is enabled.
15–9	Reserved		Reads return 0 and writes have no effect.
8	Prgm Ovrfl Ena		Program overflow enable.
		0	The program overflow exception is not enabled.
		1	The program overflow exception is enabled.
7–3	Reserved		Reads return 0 and writes have no effect.
2–0	Exception Priority Level bits		Used to select the priority of any of the three potential exception sources.
		0	The exception priority level is 2.
		1	The exception priority level is 1.

12.6.7 Exception Control Register 2 (HETEXC2)

Figure 12-42 and Table 12-17 describe the HETEXC2.

Figure 12-42. Exception Control Register 2 (HETEXC2) [offset = 0x18]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													APCNT Ovrfl flg	APCNT Undfl flg	Prgm Ovrfl flg
R-0													R/C-0	R/C-0	R/C-0

R = Read; C = Clear; *-n* = Value after reset

Table 12-17. Exception Control Register 2 (HETEXC2) Field Descriptions

Bit	Name	Value	Description
31–3	Reserved		Reads return 0 and writes have no effect.
2	APCNT Ovrfl flg		APCNT overflow flag
		0	*Read:* No exception has occurred since the flag was cleared. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* An exception has occurred since the flag was cleared. *Write:* The bit is cleared.
1	APCNT Undrfl flg		APCNT underflow flag
		0	*Read:* Exception has not occurred since the flag was cleared. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* Exception has occurred since the flag was cleared. *Write:* The bit is cleared.
0	Prgrm Overfl flg		Program overflow flag
		0	*Read:* No exception has occurred since the flag was cleared. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* An exception has occurred since the flag was cleared. *Write:* The bit is cleared.

12.6.8 Interrupt Priority Register (HETPRY)

Figure 12-43 and Table 12-18 describe the HETPRY register.

Figure 12-43. Interrupt Priority Register (HETPRY) [offset = 0x1C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HETPRY[31:16]															
RWP-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
HETPRY[15:0]															
RWP-0															

R = Read; WP = Write in privileged mode only; -*n* = Value after reset

Table 12-18. Offset Index Priority Level 2 Register (HETOFF2) Field Descriptions

Bit	Name	Value	Description
31–0	HETPRY[31:0]		HET priority level bits. These bits are used to select the priority of any of the 32 potential interrupt sources coming from the instructions.
		0	The software priority level is 2.
		1	The software priority level is 1.

12.6.9 HET Interrupt Flag Register (HETFLG)

Figure 12-44 and Table 12-19 describe the HETFLG.

Figure 12-44. HET Interrupt Flag Register (HETFLG) [offset = 0x20]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HETFLAG[31:16]															
R/C-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
HETFLAG[15:0]															
R/C-0															

R = Read; C = Clear; -*n* = Value after reset

Table 12-19. HET Interrupt Flag Register (HETFLG) Field Descriptions

Bit	Name	Value	Description
31–0	HETFLAG[31:0]		Interrupt flag. These bits are set when an interrupt condition has occurred and when the interrupt enable bit (in the instruction) is set. The flag is also set by the five LSBs of the instruction address that generated the interrupt. To clear the flag, these bits must be set to 1 or the reading of the corresponding HETOFFx register will automatically clear the flag.
		0	*Read:* No HET instruction with an interrupt has been reached since the flag was last cleared. *Write:* The bit is unchanged.
		1	*Read:* A HET instruction with an interrupt has been reached since the flag was last cleared. *Write:* The bit is cleared.

12.6.10 HR Share Control Register (HETHRSH)

Figure 12-45 and Table 12-20 describe the HETHRSH register.

Figure 12-45. HR Share Control Register (HETHRSH) [offset = 0x2C]

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

R = Read; W = Write; -*n* = Value after reset

Table 12-20. HR Share Control Register (HETHRSH) Field Descriptions

Bit	Name	Value	Description
31–12	Reserved		Reads return 0 and writes have no effect.
11–0	HR Sharex		HR share bits. These bits enable the share of the same pin for two HR structures. For example, if bit HR share 1/0 is set, the pin HET0 will then be connected to both HR structures 0 and 1.
		0	Pins are not shared.
		1	Pins are shared.
			If HR share bits are used, pins not connected to HR structures can be accessed as general inputs/outputs

12.6.11 HR XOR-Share Control Register (HETXOR)

Figure 12-46 and Table 12-21 describe the HETXOR register.

Figure 12-46. HR XOR-Share Control Register (HETXOR) [offset = 0x30]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				HR XOR Share 23/22	HR XOR Share 21/20	HR XOR Share 19/18	HR XOR Share 17/16	HR XOR Share 15/14	HR XOR Share 13/12	HR XOR Share 11/10	HR XOR Share 9/8	HR XOR Share 7/6	HR XOR Share 5/4	HR XOR Share 3/2	HR XOR Share 1/0
R-0				R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

R = Read; W = Write; *-n* = Value after reset

Table 12-21. HR XOR-Share Control Register (HETXOR) Field Descriptions

Bit	Name	Value	Description
31–12	Reserved		Reads return 0 and writes have no effect.
11–0	HR XOR-Sharex		HR XOR-share bits These bits enable the XOR-share of the same pin for two HR structures. For example, if bit HR XOR-share 1/0 is set, the pin HET0 will then be commanded by a logical XOR of both HR structures 0 and 1.
		0	Pins are not XOR-shared.
		1	Pins are XOR-shared. If HR share bits are used, pins not connected to HR structures can be accessed as general inputs/outputs

12.6.12 HET Direction Register (HETDIR)

Figure 12-47 and Table describe the HETDIR register.

Figure 12-47. HET Direction Register (HETDIR) [offset = 0x34]

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

R = Read; W = Write; *-n* = Value after reset

Table 12-22. HET Direction Register (HETDIR) Field Descriptions

Bit	Name	Value	Description
31–0	HET DIR[31:0]		Input/output direction These bits are used to select the direction of the HET pins.
		0	HET pin configured as input.
		1	HET pin configured as output.

12.6.13 HET Data Input Register (HETDIN)

Figure 12-48 and Table 12-23 describe the HETDIN register.

Figure 12-48. HET Data Input Register (HETDIN) [offset = 0x38]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HETDIN[31:16]															
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HETDIN[15:0]															
R-0															

R = Read; *-n* = Value after reset

Table 12-23. HET Data Input Register (HETDIN) Field Descriptions

Bit	Name	Value	Description
31–0	HET DIN[31:0]		HET data input. Read operation mode (writes have no effect):
		0	The pin is at logic low (0).
		1	The pin is at logic high (1).

12.6.14 HET Data Output Register (R-Write) (HETDOUT)

Figure 12-49 and Table 12-24 describe the HETDOUT register.

Figure 12-49. HET Data Output Register (R-Write) (HETDOUT) [offset = 0x3C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HETDOUT[31:16]															
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HETDOUT[15:0]															
R/W-0															

R = Read; W = Write; *-n* = Value after reset

Table 12-24. HET Data Output Register (R-Write) (HETDOUT) Field Descriptions

Bit	Name	Value	Description
31–0	HETDOUT[31:0]		HET data output. This bit is used to set/reset the HET pins. A read to this register gives the value of the corresponding HETDSET register (Section 12.6.14).
		0	The pin is at logic low (0).
		1	The pin is at logic high (1).

12.6.15 HET Data Set Register (R-Set) (HETDSET)

Figure 12-50 and Table 12-25 describe the HETDSET register.

Figure 12-50. HET Data Set Register (R-Set) (HETDSET) [offset = 0x40]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HETDSET[31:16]															
RS-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HETDSET[15:0]															
RS-0															

R = Read; S = Set; *-n* = Value after reset

Table 12-25. HET Data Set Register (R-Set) (HETDSET) Field Descriptions

Bit	Name	Value	Description
31–0	HETDSET[31:0]		HET data set. These bits are used to set the HET pins to the HIGH level. A read of HETDSET reflects the contents of the HETDOUT register (Section 12.6.14).
		0	*Read:* The pin is at logic low (0). *Write:* The corresponding bit in HETDOUT is unchanged.
		1	*Read:* The pin is at logic high (1). *Write:* The corresponding bit in HETDOUT is set to 1.

12.6.16 HET Data Clear Register (R-Clear) (HETDCLR)

Figure 12-51 and Table 12-26 describe the HETDCLR register.

Figure 12-51. HET Data Clear Register (R-Clear) (HETDCLR) [offset = 0x44]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HETDCLR[31:16]															
R/C-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
HETDCLR[15:0]															
R/C-0															

R = Read; C = Clear; -*n* = Value after reset

Table 12-26. HET Data Clear Register (R-Clear) (HETDCLR) Field Descriptions

Bit	Name	Value	Description
31–0	HETDCLR[31:0]		HET data clear. These bits are used to clear the HET pins to 0. A read of HETDCLR reflects the contents of HETDOUT (Section 12.6.14).
		0	*Read:* The pin is at logic low (0). *Write:* The corresponding bit in HETDOUT is unchanged.
		1	*Read:* The pin is at logic high (1). *Write:* The corresponding bit in HETDOUT is set to 1.

12.6.17 HET Open Drain Register (HETPDR)

Figure 12-52 and Table 12-27 describe this register.

Figure 12-52. HET Open Drain Register (HETPDR) [offset = 0x48]

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

R = Read; W = Write; -*n* = Value after reset

Table 12-27. HET Open Drain Register (HETPDR) Field Descriptions

Bit	Name	Value	Description
31–0	HETPDR[31:0]		HET open drain.
		0	The open drain function is disabled (the output voltage is V_{OL} or lower if HETDOUT = 0, and V_{OH} or higher if HETDOUT = 1).
		1	The open drain function is enabled (the output voltage is V_{OL} or lower if HETDOUT = 0, and Z if HETDOUT = 1).

12.6.18 HET Pull Disable Register (HETPULDIS)

This register enables or disables the pull control function of the pins. Figure 12-53 and Table 12-28 describe this register.

Figure 12-53. HET Pull Disable Register (HETPULDIS) [offset = 0x4C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HETPULDIS[31:16]															
R/W-x															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
HETPULDIS[15:0]															
R/W-x															

R = Read; W = Write; -*x* = Device specific tie off after reset (please read device-specific pin description)

Table 12-28. HET Pull Disable Register (HETPULDIS) Field Descriptions

Bit	Name	Value	Description
31–0	HETPULDIS[31:0]		HET pull disable.
		0	The pull function is enabled.
		1	The pull function is disabled.

12.6.19 HET Pull Select Register (HETPSL)

This register selects the pull up or pull down function of the pins. Figure 12-54 and Table 12-29 describe this register.

Figure 12-54. HET Pull Select Register (HETPSL) [offset = 0x50]

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

R = Read; W = Write; -*n* = Value after reset

Table 12-29. HET Pull Select Register (HETPSL) Field Descriptions

Bit	Name	Value	Description
31–0	HETPSL[31:0]		HET pull select
		0	The pull down function is enabled.
		1	The pull up function is enabled.

12.6.20 HET Loopback Pair Select Register (HETLPBSEL)

This register selects whether the loopback mode on the dedicated pins should be enabled and analog or digital loopback can be configured. Figure 12-55 and Table 12-30 describe this register.

Figure 12-55. HET Loopback Pair Select Register (HETLPBSEL) [offset = 0x60]

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

R = Read; W = Write; -*n* = Value after reset

Table 12-30. HET Loopback Pair Select Register (HETLPBSEL) Field Descriptions

Bit	Name	Value	Description
31–16	HETLPBTYPE[2x+1/2x]		Loopback type select
		0	Digital loopback is selected.
		1	Analog loopback is selected.
15–0	HETLPBSEL[2x+1/2x]		Loopback pair select
		0	No loopback mode selected.
		1	High resolution structures of pin 2x and 2x+1 are connected together. The direction is given by the HETLPBDIR register.

12.6.21 HET Loopback Pair Direction Register (HETLPBDIR)

This register selects the direction of the loopback mode on the dedicated pins. The loopback direction is independent of the HETDIR register setting. Figure 12-56 and Table 12-31 describe this register.

Figure 12-56. HET Loopback Pair Direction Register (HETLPBDIR) [offset = 0x64]

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

R = Read; W = Write; *-n* = Value after reset

Table 12-31. HET Loopback Pair Direction Register (HETLPBDIR) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Reads return 0 and writes have no effect.
15–0	HETLPBDIR[2x+1/2x]		Loopback direction select.
		0	HR structure 2x is input and 2x+1 is output.
		1	HR structure 2x+1 is input and 2x is output.

12.6.22 HET Parity Control Register (HETPCR)

Figure 12-57 and Table 12-32 describe the HETPCR register.

Figure 12-57. HET Parity Control Register (HETPCR) [offset = 0x68]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															HET-STOP
R-0															R/WP-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							TEST	Reserved				PARITY_ENA			
R-0							R/WP-0	R-0				R/WP-0101			

R = Read; WP = Write in privilege mode only; -*n* = Value after reset

Table 12-32. HET Parity Control Register (HETPCR) Field Descriptions

Bit	Name	Value	Description
31–17	Reserved		Reads return 0 and writes have no effect.
16	HET_STOP		If this bit is set the HET will stop execution when a parity error is detected. This is done by automatically turning off the HET_ON bit in the HETGCR register. User and Privilege mode (read): 0 HET will not stop execution in the case of a parity error 1 HET will stop execution in the case of a parity error Privilege mode (write): 0 HET will not stop execution in the case of a parity error 1 HET will stop execution in the case of a parity error
15–9	Reserved		Reads return 0 and writes have no effect.
8	TEST		This bit maps the parity bits into the peripheral RAM frame to make them accessible by the CPU. User and Privilege mode (read): 0 Parity bits are not memory mapped 1 Parity bits are memory mapped Privilege mode (write): 0 Disable mapping 1 Enable mapping
7–4	Reserved		Reads return 0 and writes have no effect.

Table 12-32. HET Parity Control Register (HETPCR) Field Descriptions (Continued)

Bit	Name	Value	Description
3–0	PARITY_ENA		This bit 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 HET_UERR signal is activated. User and Privilege mode (read): 0101 Parity check is disabled all other Parity check is enabled Privilege mode (write): 0101 Disable checking Enable checking all other NOTE: It is recommended to write “1010” to enable error detection, to guard against soft error from flipping EDEN to a disable state.

12.6.23 HET Parity Interrupt Enable Register (HETPIEN)

Figure 12-58 and Table 12-33 describe the HETPIEN register.

Figure 12-58. HET Parity Interrupt Enable Register (HETPIEN) [offset = 0x6C]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															INT EN
R-0															RW-0

R = Read; W = Write; -*n* = Value after reset

Table 12-33. HET Parity Interrupt Enable (HETPIEN) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved		Reads return 0 and writes have no effect.
0	INTEN		Interrupt Enable
		0	HET UERR interrupt is disabled
		1	HET UERR interrupt is enabled

12.6.24 HET Parity Interrupt Flag Register (HETPIFLG)

Figure 12-59 and Table 12-34 describe the HETPIFLG register.

Figure 12-59. HET Parity Interrupt Flag Register (HETPIFLG) [offset = 0x70]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															INT FLG
R-0															RC-0

R = Read; *-n* = Value after reset

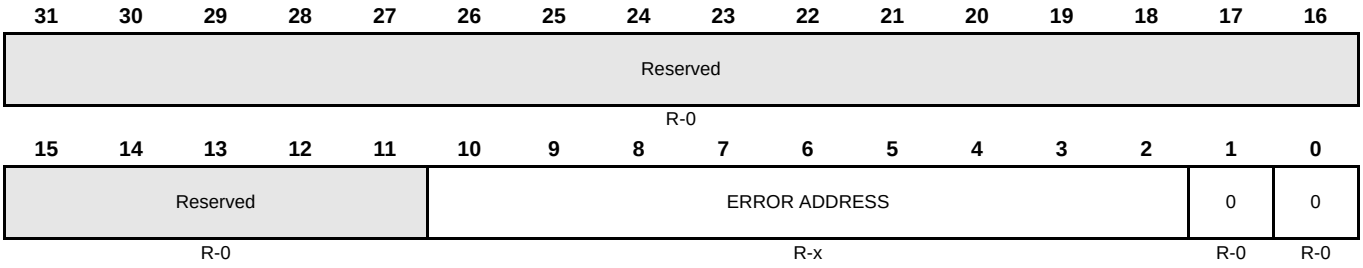
Table 12-34. HET Parity Interrupt Flag (HETPIFLG) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved		Reads return 0 and writes have no effect.
0	INTFLG		Interrupt Flag Writes have no effect
		0	No parity error has occurred
		1	Parity error has occurred This bit is cleared if a 1 is written to it. It is also cleared when HETPAR register is read.

12.6.25 HET Parity Address Register (HETPAR)

Figure 12-60 and Table 12-35 describe the HETPAR register.

Figure 12-60. HET Parity Address Register (HETPAR) [offset = 0x74]



R = Read; -*n* = Value after reset

Table 12-35. HET Parity Address Register (HETPAR) Field Descriptions

Bit	Name	Value	Description
31–11	Reserved		Reads return 0 and writes have no effect.
10–2	ERROR ADDRESS		This register holds 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.
1–0	0 (Read only)	0	These bits read as 0, to allow for reading the word aligned address of the failure.

12.7 Instruction Set

Table 12-36 presents a list of the instructions in the HET instruction set. Table 12-37 presents the flags generated by the instructions, and Table 12-38 lists the instructions as to whether they are interrupt capable or not. The following sections discuss the HET instruction set in detail.

Table 12-36. Instruction Summary

Abbreviation	Instruction Name	Opcode	Sub-Opcode	Cycles	Reference
ACMP	Angle Compare	Ch	-	1	Page 661
ACNT	Angle Count	9h	-	2	Page 663
ADCNST	Add Constant	5h	-	2	Page 666
ADM32	Add Move 32	4h	C5=1	1 or 2	Page 667
APCNT	Angle Period Count	Eh	-	1 or 2	Page 671
BR	Branch	Dh	-	1	Page 674
CNT	Count	6h	-	1 or 2	Page 676
DADM64	Data Add Move 64	2h	-	2	Page 679
DJZ	Decrement and Jump if -zero	Ah	P[6-5]=10	1	Page 681
ECMP	Equality Compare	0h	C[6-5]=00	1	Page 683
ECNT	Event Count	Ah	P[6-5]=01	1	Page 686
MCMP	Magnitude Compare	0h	C[6-5]=1X	1	Page 688
MOV32	Move 32	4h	C5=0	1 or 2	Page 691
MOV64	Move 64	1h	-	1	Page 695
PCNT	Period/Pulse Count	7h	-	1	Page 697
PWCNT	Pulse Width Count	Ah	P[6-5]=11	1	Page 700
RADM64	Register Add Move 64	3h	-	1	Page 702
SCMP	Sequence Compare	0h	C[6-5]=01	1	Page 704
SCNT	Step Count	Ah	P[6-5]=00	3	Page 707
SHFT	Shift	Fh	-	1	Page 709
WCAP	Software Capture Word	Bh	-	1	Page 712

Table 12-37. FLAGS Generated by Instruction

	Flag Name	Set / Reset by	Used by
Z	Z flag	CNT, APCNT, PCNT, ACNT, SCNT, SHFT	ACMP, ECMP, SCMP, MCMP, ACNT, BR, SHFT
X	X flag	ACMP	SCMP
SWF 0-1	Step width flags	SCNT	ACNT
NAF	New angle flag	ACNT	NAF_global
NAF_global	New angle flag (global)	NAF	ACNT, CNT, ECNT, BR, ECMP, MCMP
ACF	Acceleration flag	ACNT	SCNT, ACNT
DCF	Deceleration flag	ACNT	SCNT, ACNT

	Flag Name	Set / Reset by	Used by
GPF	Gap flag	ACNT	APCNT, ACNT

The instructions capable of generating SW interrupts are listed in Table 12-38.

Table 12-38. Interrupt-Capable Instructions

Interrupt-Capable Instructions	Non-Interrupt-Capable Instructions
ACMP	ADCNST
ECMP	ADM32
SCMP	DADM32
MCMP	MOV32
CNT	MOV64
ECNT	RADM64
ACNT	SCNT
APCNT	
PWCNT	
PCNT	
DJZ	
WCAP	
SHFT	
BR	

12.7.1 Abbreviations

Abbreviations marked with a star (*) are available only on specific instructions.

BRK	Defines the software breakpoint for the device software debugger. Default: OFF. Location: Program field [20]
Next	Defines the program address of the next instruction in the program flow. This value may be a label or an 8-bit unsigned integer. Default: Current instruction + 1. Location: Program field [19:12]
Remote(1)	Determines the 8-bit address of the remote address for the instruction. Default: Current instruction + 1. Location: Program field [7:0] **Note : *HET does not support next address equal to remote address for any instruction which is capable of doing data transfer***
Control	Determines whether the immediate data field [24:0] 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 [21]
En_pin_action(1)	Determines whether the selected pin is ON so that the action occurs on the chosen pin. Default: OFF. Location: Control field [20]
Cond_addr(1)	Conditional address: (Optional) Defines the address of the next instruction when the condition occurs. Default: Current address + 1. Location: Control field [19:12]
Pin(1)	Pin Select: Selects the pin on which the action occurs. Enter the pin number.(1) Default: pin 0. Location: Control field [11:7] except PCNT,
U	Reading a bit marked with U will return an indeterminate value.

1 The format CC{pin number} is also supported.

12.7.2 Encoding Formats and Bits

Table 12-39 through Table 12-44 provide information about encoding formats and bits.

Table 12-39. PIN Encoding Format

MSB				LSB	Pin Select
0	0	0	0	0	Selects HET0
0	0	0	0	1	Selects HET1
(each pin may be selected by writing its number in binary)					
1	1	1	1	0	Selects HET30
1	1	1	1	1	Selects HET31

Reg(1)	Register select: Selects the register for data comparison and storage. Default: No register (none). Location: Control field [2:1] except CNT

1 The format CC{pin number} is also supported.

Table 12-40. Register Bit Field Encoding Format

Register[1]	C[2]	C[1]
A	0	0
B	0	1
T	1	0
None	1	1

1 The register bits field could be placed either in the program field (CNT), or in the control field (all others' instructions use register field).

Action[1] (Two action option) Either sets or clears the pin
Default: Clear.
Location: Control field [4]

1 The format CC{pin number} is also supported.

Table 12-41. PIN Action Bit Field (2 options)

Action	C[4]
Clear	0
Set	1

Action[1] (Four-action option) Either sets, clears, pulse high or pulse low on the pin. Pulse high occurs when the pin is set on the compare and toggles at the overflow.
Default: Clear.
Location: Control field [4:3]

1 The format CC{pin number} is also supported.

Table 12-42. PIN Action Bit Field (4 options)

Action	C[4][1]	C[3][2]
Clear	0	0
Set	1	0
Pulse Low	0	1
Pulse High	1	1

1 Bit C[4] is also called enable pin action.
2 C[3] is also called opposite pin action.

hr_lr[1] Specifies high/low data resolution. If the hr_lr field is high, the instruction implements the hr_data field (when the action is carried out on an HR pin). If the hr_lr field is low, the hr_data field is ignored.

Default: High.
Location: Program field [7]

1 The format CC{pin number} is also supported.

Table 12-43. High-Low Resolution Bit Field

hr_lr	Prog. field [7]
Low	1
High	0

prv(1) Specifies the initial value defining the previous pin-level bit for the first edge detect performed by the instruction. The edge detect is performed by comparing the current pin value to the value stored in the previous pin-level bit. 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. After the initial comparison, the value of the prv bit is set or reset by the system.
Default: OFF.
Location: Control field [20]

cntl_val(1) Available for the DADM64, MOV64, and RADM64, this bits field allows you to specify the replacement value for the remote control field.

comp_mode(1) Specifies the compare mode. This field is used with the 64-bit move instructions. The field ensures that the sub-opcodes are moved correctly.
Default: ECMP.
Location: Control field [6:5]

1 The format CC{pin number} is also supported.

Table 12-44. Comp_mode Bit Field

comp_mode	C[6]	C[5]
ECMP	0	0
SCMP	0	
MCMP	1	
ACMP	1	

12.7.3 Instruction Description

The following sections provide the details for each instruction.

12.7.3.1 ACMP (Angle Compare)

Syntax

```
ACMP {
[brk={OFF | ON}]
[next={label | 8-bit unsigned integer}]
[control={OFF | ON}]
[en_pin_action={OFF | ON}]
[cond_addr={label | 8-bit unsigned integer}]
pin={pin number}
[action={CLEAR | SET}]
reg={A | B | T | NONE}
[irq ={OFF | ON}]
data={20-bit unsigned integer}
}
```

Opcode — Ch, [P11:P8]

Table 12-45. ACMP Program Field (P31:P0)

31–22	21	20	19–12	11–8	7–0
0	Res	BRK	Next program address	Opcode	Reserved
10	1-U	1	8	4	8

Table 12-46. ACMP Control Field (C31:C0)

31–22	21	20	19–12	11–7	6	5	4	3	2–1	0
0	Control	En. pin action	Conditional address	Pin select	Res.	Cout prv	Pin action	Res.	Register select	Int. ena.
10	1	1	8	5	1	1	1	1	1	1

Table 12-47. ACMP Data Field (D31:D0)

31–25	24–5	4–0
0	Data	Reserved
7	20	5

Cycles — One

Register modified — Selected register (A, B, or T)

Description — 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 20-bit angle compare value.

Execution

```
X = 0;
If (Data field value ≤ Selected register value)
    Cout = 0;
else
    Cout = 1;
If (Z == 0 AND (Selected register value - Angle Inc. < Data field
value) 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)
          SW interrupt flag = 1;
    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 4-bit input.

12.7.3.2 ACNT (Angle Count)

Syntax

```
ACNT {
[brk={OFF | ON}]
[next={label | 8-bit unsigned integer}]
edge={RISING | FALLING}
[irq={OFF | ON}]
[control={OFF | ON}]
[prv={OFF | ON}]
gapend=20-bit unsigned integer
[data=20-bit unsigned integer]
}
```

Opcode 9h, [P11:P8]

Table 12-48. ACNT Program Field (P31:P0)

31–22	21	20	19–12	11–8	7	6–1	0
0	Res	BRK	Next program address	Opcode	Edge-select.	Reserved	Int. ena.
10	1-U	1	8	4	1	6	1

Table 12-49. ACNT Control Field (C31:C0)

31–22	21	20	19–0
0	Control	Prv.	Gap End
10	1	1	20

Table 12-50. ACNT Data Field (D31:D0)

31–25	24–5	4–0
0	Data	Reserved
7	20	5

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 HET24 exclusively**.** The edge select must be the same as the HET24 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[24]).

Action	P7	Edge Select
Rising	1	Detects a rising edge of HET24
Falling	0	Detects a falling edge of HET24

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 20-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 20-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 HET24
Target Edge Condition: (Target Edge field in data field = 0) AND (An-
gle Increment condition is true) AND (GPF = 0)

If (Angle Increment Condition) is false
    {
    NAF_global = 0;
    Register B = Data field register;
    }

else
    {
    NAF_global = 1;
    If (Counter value != GapEnd - 1)
          {
          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 HET24) AND (data
          field register != 0) AND (ACF == 0))
            DCF = 1;
          }
    If (specified edge is detected on pin HET24)
          {
          DCF = 0;
          If ((data field register != 0) AND (DCF == 0))
            ACF = 1;
          If (GPF == 1)
            {
            GPF = 0;
            Z = 1;
            If (Interrupt Enable == 1)
              SW interrupt flag = 1;
            }
          }
    }
Prv bit = Current HET24 pin level;
```

12.7.3.3 ADCNST (Add Constant)

Syntax

ADCNST {
[brk={OFF | ON}]
[next={*label* | *8-bit unsigned integer*}]
[control={OFF | ON}]
remote={*label* | *8-bit unsigned integer*}
min_off=*20-bit unsigned integer*
data=*20-bit unsigned integer*
[hr_data=5-*bit unsigned integer*]
}

Opcode

5h, [P11:P8]

Table 12-51. ADCNST Program Field (P31:P0)

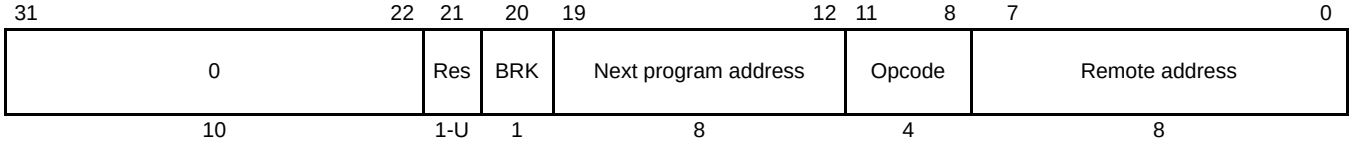


Table 12-52. ADCNST Control Field (C31:C0)

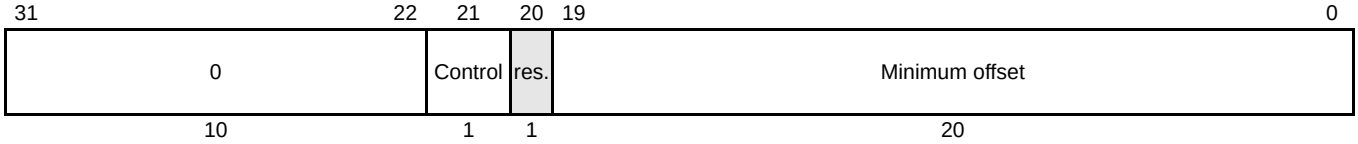
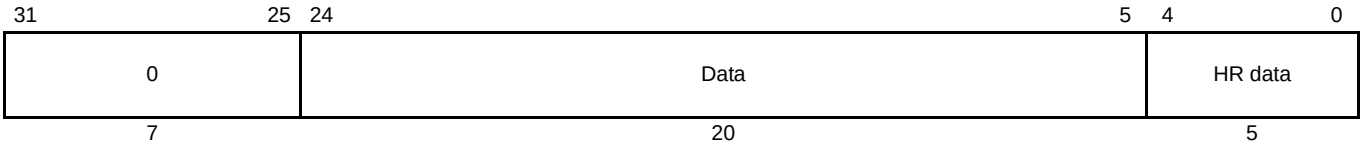


Table 12-53. ADCNST Data Field (D31:D0)



Cycles

Two

Register Modified

Register T (implicitly)

Description

ADCNST is an extension of ADMOV32. 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 20-bit constant value that is added to the data field value if the remote data field is null.

data — A 20-bit value that is always added to the remote data field.
Default: 0.

hr_data — Five least significant bits of the data addition to the remote data field.
Default: 0.

Table 12-61 and Table 12-62 illustrate the behavior of ADCNST if the remote data field is or is not zero.

Figure 12-61. ADCNST Operation If Remote Data Field[24:5] Is Not Zero

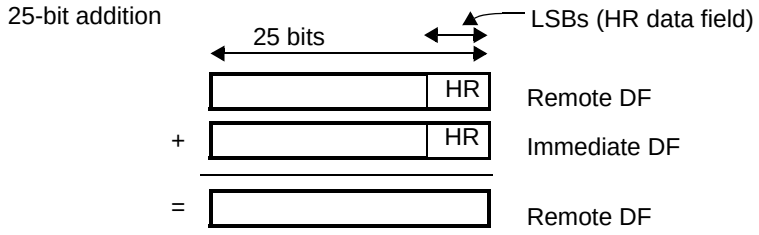
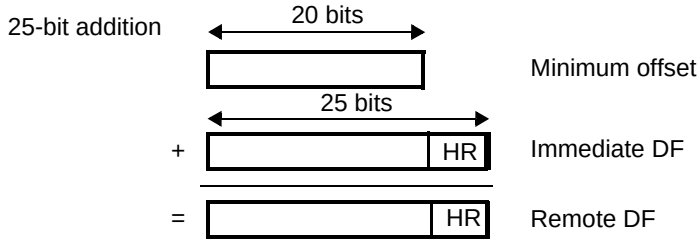


Figure 12-62. ADCNST Operation if Remote Data Field [24:5] Is Zero




```
Execution If (Remote Data Field Value [24:5] != 0)
              Remote Data Field = Immediate Data Field + Remote Data Field;
          else
              Remote Data Field = Immediate Data Field + min. offset(bits
              C19:C0);
          Jump to Next Program Address;
```

12.7.3.4 ADM32 (Add Move 32)

Syntax

ADM32 {
[brk={OFF | ON}]
[next={*label* | *8-bit unsigned integer*}]
remote={*label* | *8-bit unsigned integer* }
[control={OFF | ON}]
[init={OFF | ON}]
type={IM&REGTOREG | REM&REGTOREG |
IM&REMTOREG | IM&REGTOREM}
reg={A | B | T | NONE}
data=*20-bit unsigned integer*
[hr_data=5-*bit unsigned integer*]
}

Opcode — 4h, [P11:P8];

Sub-Opcode — *1h, [C5]*

Table 12-54. ADM32 Program Field (P31:P0)

31 — 22	21	20	19 — 12	11 — 8	7 — 0
0	Res	BRK	Next program address	Opcode	Remote address
10	1-U	1	8	4	8

Table 12-55. ADM32 Control Field (C31:C0)

31 – 22	21	20 – 7	6	5	4 – 3	2 – 1	0
0	Control	Reserved	Init flag	Sub-opcode	Move type	Register	Res.
10	1	14	1	1	2	2	1

Table 12-56. ADM32 Data Field (D31:D0)

31 – 25	24 – 5	4 – 0
0	Data field	HR data
7	20	5

Cycles — One or two cycles (see Table 12-57)

Register Modified — Selected register (A, B, 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 12-57. Move Types for ADM32

Type	C4	C3	Add			Destination(s)	Cycles
IM&REGTOREG	0	0	Imm. data field	+	Reg. A, B, or T	Register A, B, or T	1
REM&REGTOREG	0	1	Remote data field	+	Reg. A, B, or T	Register A, B, or T	2
IM&REMTOREG	1	0	Imm. data field	+	Remote data field	Register A, B, or T	2
IM&REGTOREM	1	1	Imm. data field	+	Reg. A, B, or T	Remote data field	1

If selected register is T, the operation is a 25-bit Addition/move. If A or B register is selected, it is limited to 20-bit operation since A and B only support 20-bit.

data — Specifies the 20-bit integer value for the immediate data field.

hr_data — Specifies the 5 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;
```

Register Modified — Selected register (A, B or T)

Figure 12-63 through Figure 12-66 illustrate the ADM32 operation for various cases.

Figure 12-63. ADM32 Add and Move Operation for IM&REGTOREM (Case 00)

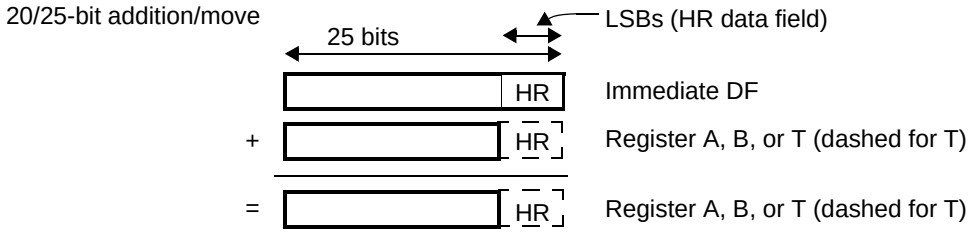


Figure 12-64. ADM32 Add and Move Operation for REM&REGTOREG (Case 01)

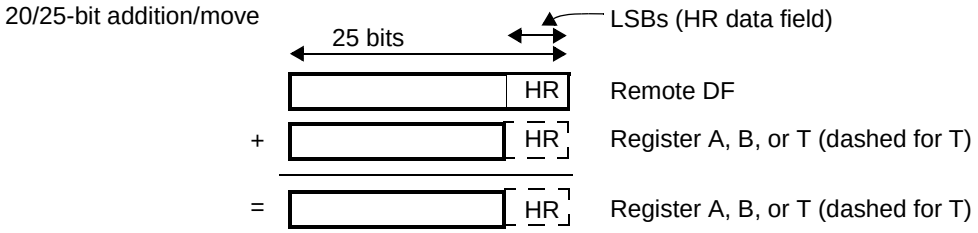


Figure 12-65. ADM32 Add and Move Operation for IM&REMTOREG (Case 10)

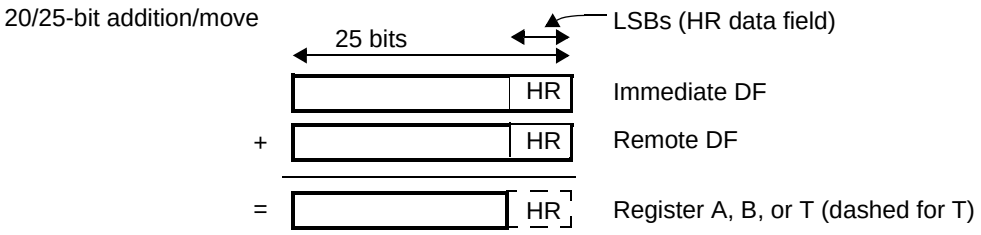
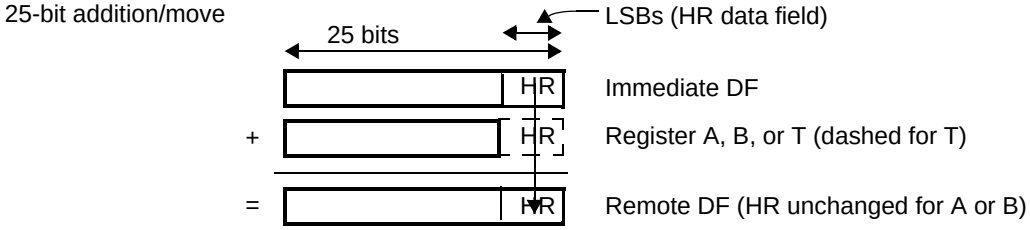


Figure 12-66. ADM32 Add and Move Operation for IM&REGTOREM (Case 11)



12.7.3.5 APCNT (Angle Period Count)

Syntax

```
APCNT {
[brk={OFF| ON}]
[next={label | 8-bit unsigned integer}]
[irq={OFF | ON}]
type={FALL2FALL | RISE2RISE}
[control={OFF | ON}]
prv={OFF | ON}
period=20-bit unsigned integer
[data=20-bit unsigned integer]
}
```

Opcode Eh, [P11:P8]

Table 12-58. APCNT Program Field (P31:P0)

31–22	21	20	19–12	11–8	7	6–5	4–0
0	Res	BRK	Next program address	Opcode	Int. ena.	Edge select	Reserved
10	1-U	1	8	4	1	2	5

Table 12-59. APCNT Control Field (C31:C0)

31–22	21	20	19–0
0	Control	Prv.	Period count
10	1	1	20

Table 12-60. APCNT Data Field (D31:D0)

31–25	24–5	4–0
0	Data	Reserved
7	20	5

Cycles One or two cycles
One cycle (normal operation) two cycles (edge detected)

- Cycle 1: edge detected (normal operation)
- Cycle 2: edge detected and GPF = 1 and underflow condition is true

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 HET24. The toothed wheel must then be connected to pin HET24.

APCNT uses the gap flag (GPF) defined by ACNT to start or stop captures in the period count field [C19: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 12-61.

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 12-61. Edge Select Encoding for APCNT

type	P6	P5	Selected Condition
Fall2Fall	1	0	Falling edge
Rise2Rise	1	1	Rising edge

period — Contains the 20-bit count value from the previous APCNT period.

data — 20-bit value serving as a counter.
Default: 0.

Execution

```
Z = 0;
If (Data field register != FFFFFh)
    {
    Register A = Data field register + 1;
    Data field register = Data field register + 1;
    }
elseIf (specified edge not detected on HET24)
    {
    Register A = FFFFFh;
    APCNT Ovflw flag = 1;
    }
If (specified edge detected on HET24)
    {
    Z = 1;
    If (Data field register == FFFFFh)
          {
          Register A = FFFFFh;
          Register T = FFFFFh;
          Period count = FFFFFh;
          }
    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)
                              SW interrupt flag = 1;
                            }
                        If (GPF == 1)
                            Register T = Period count;
    If (Data Field register < Step width)
                            {
                            Register T = Period count;
                            APCNT Undflw flag = 1;
                            Data field register = 00000h;
                            }
                        }
                    else
                        Register T = Period count;
                    Prv bit = Current HET24 pin level;
                    Jump to Next Program Address;
```

12.7.3.6 BR (Branch)

Syntax

```
BR {
[brk={OFF | ON}]
[next={label | 8-bit unsigned integer}]
[control={OFF | ON}]
[prv={OFF | ON}]
cond_addr={label | 8-bit unsigned integer}
[pin= {pin number}]
event={NOCOND | FALL | RISE | BOTH | ZERO | NAF | LOW | HIGH}
[irq={OFF | ON}]
}
```

Opcode: Dh, [P11:P8]

Table 12-62. BR Program Field (P31:P0)

31–22	21	20	19–12	11–8	7–0
0		BRK	Next program address	Opcode	Reserved
10	1-U	1	8	4	8

Table 12-63. BR Control Field (C31:C0)

31–22	21	20	19–12	11–7	6–4	3–1	0
0	Control	Prv.	Conditional address	Pin select	Branch cond.	Reserved	Int. ena.
10	1	1	8	5	3	3	1

Table 12-64. BR Data Field (D31:D0)

31–25	24–0
0	Reserved
7	25

Cycles: One

Register Modified: None

Description: This instruction executes a jump to the conditional address [C19:C12] on a pin or a flag condition and can be used with all pins.

An edge is detected by comparing the current pin level sampled at loop start to the previous pin level stored in the Prv. (Previous) bit [C20].

event: Specifies the event that triggers a jump to the indexed program address.
Default: FALL

Table 12-65 provides the branch condition encoding.

Table 12-65. Branch Condition Encoding for BR

event	C6	C5	C4	Branch Condition
NOCOND	0	0	0	Always
FALL	0	0	1	On falling edge on the selected pin
RISE	0	1	0	On rising edge on selected pin
BOTH	0	1	1	On rising or falling edge on selected pin

event	C6	C5	C4	Branch Condition
ZERO	1	0	0	If Zero flag is set
NAF	1	0	1	If NAF_global flag is set
LOW	1	1	0	On LOW level on selected pin
HIGH	1	1	1	On HIGH level on selected pin

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.

Execution

```
If (Condition is true)
    {
    If (Interrupt Enable == 1)
          SW interrupt flag = 1;
    Jump to Conditional Address;
    }
else
    Jump to Next Program Address;
Prv. bit = Selected Pin level; (Always executed)
```

12.7.3.7 CNT (Count)

Syntax

```
CNT {
[brk={OFF | ON}]
[next={label | 8-bit unsigned integer}]
[angle_count={OFF | ON}]
[reg={A | B | T | NONE}]
[irq={OFF | ON}]
[control={OFF | ON}]
max=20-bit unsigned integer
[data=20-bit unsigned integer]
}
```

Opcode 6h, [P11:P8]

Table 12-66. CNT Program Field (P31:P0)

31 22	21	20	19 12	11 8	7	6 5	4 1	0
0	Res	BRK	Next program address	Opcode	Angle count	Register	Reserved	Int. ena.
10	1-U	1	8	4	1	2	4	1

Table 12-67. CNT Control Field (C31:C0)

31 22	21	20	19 0
0	Control	res.	Max count
10	1	1	20

Table 12-68. CNT Data Field (D31:D0)

31 25	24 5	4 0
0	Data field	Reserved
7	20	5

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 [D24:5] is incremented unconditionally on each resolution when in time mode (angle count bit [P7] = 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 [P7] = 1), CNT needs data from the software angle generator (SWAG). For the SWAG, the angle increment value will be 0 or 1. It takes two cycles in this mode.

Table 12-40 shows the P6 and P5 encoding for selecting the required ALU register

.

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 assembler encounters the CNT instruction. Default value for this field is OFF.
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 20-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 20-bit integer value serving as a counter. Default: 0.

Execution

```
Z = 0;
If (Angle Count (bit P7 == 1))
    {
    If (NAF_global == 0)
          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)
              SW interrupt flag = 1;
            }
          else
            {
            Selected register = Immediate Data Field + Angle Incre-
            ment;
            Immediate Data Field = Immediate Data Field + Angle Incre-
            ment;
            }
          }
    }
else (Time mode (bit P7 == 0))
    {
    If (Immediate Data Field == Max count)
          {
          Z = 1;
          Selected register = 00000;
```

```
                Immediate Data Field = 00000;
                If (Interrupt Enable == 1)
                  SW interrupt flag = 1;
                }
            else
                {
                Selected register = Immediate Data Field + 1;
                Immediate Data Field = Immediate Data Field + 1;
                }
            }
        Jump to Next Program Address;
```

12.7.3.8 DADM64 (Data Add Move)

Syntax

DADM64 {
[brk={OFF | ON}]
[next=*{label | 8-bit unsigned integer}]*
remote=*{label | 8-bit unsigned integer}*
[control={OFF | ON}]
[en_pin_action={OFF | ON}]
[cond_addr=*{label | 8-bit unsigned integer}]*
[pin=pin number]
comp_mode={ECMP | SCMP | MCMP|ACMP}
[action={CLEAR | SET | PULSELO | PULSEHI}]
reg={A | B | T | NONE}
[irq={OFF | ON}]
data=*20-bit unsigned integer*
[hr_data=5-*bit unsigned integer]*
}

-or-

Syntax

DADM64 {
[brk={OFF | ON}]
[next=*{label | 8-bit unsigned integer}]*
remote=*{label | 8-bit unsigned integer}*
[control={OFF | ON}]
cntl_val=*21-bit unsigned integer*
data=*20-bit unsigned integer*
[hr_data=*5-bit unsigned integer]*
}

Opcode *2h, [P11:P8]*

Table 12-69. DADM64 Program Field (P31:P0)

31 – 22	21	20	19 – 12	11 – 8	7 – 0
0	Res	BRK	Next Program Address	Opcode	Remote Address
10	1-U	1	8	4	8

Table 12-70. DADM64 Control Field (C31:C0)

31 – 22	21	20	19 – 12	11 – 7	6 – 5	4 – 3	2 – 1	0
0	Control	Enable Pin Action	Conditional Address	Pin Select	Comp. Mode	Action	Register	Int. Ena.
10	1	1	8	5	2	2	2	1

Table 12-71. DADM64 Data Field (D31:D0)

31 – 25	24 – 5	4 – 0
0	Data Field	High Res. Data Field
7	20	5

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 21-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 12-67 shows the DADM64 add and move operation.

Figure 12-67. DADM64 Add and Move Operation

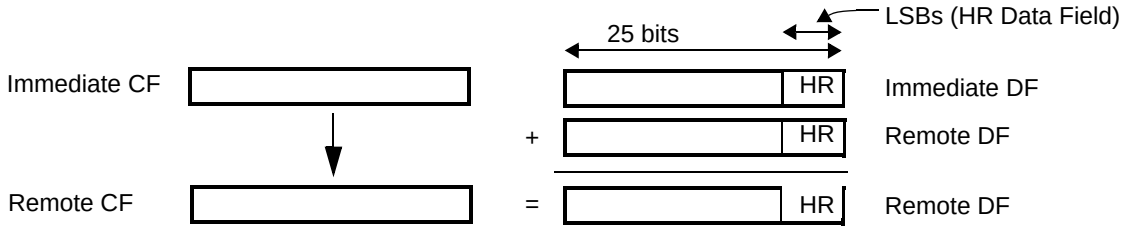


Table 12-72. DADM64 Control Field Description

Control Field	Description
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 20-bit initial value for the data field.
hr_data	Five least significant bits of the 25 bit data field. Default: 0
cntl_val	Specifies the 21 least significant bits of the Control field.

Execution

```
Remote Data Field = Remote Data Field + Immediate Data Field;
Remote Control Field = Immediate Control Field;
Jump to Next Program Address;
```

12.7.3.9 DJZ (Decrement and Jump if Zero)

Syntax[1]

DJZ{
[brk={OFF | ON}]
[next=*{label | 8-bit unsigned integer}]*
[control={OFF | ON}]
cond_addr=*{label | 8-bit unsigned integer}*
[reg={A | B | T | NONE}]
[irq={OFF | ON}]
[data=*20-bit unsigned integer]*
}

Opcode: Ah, [P11:P8];

Sub-opcode: *[P6-P5]=10*

Table 12-73. DJZ Program Field (P31:P0)

31–22	21	20	19–12	11–8	7	6–5	4–0
0	Res	BRK	Next program address	Opcode	Res.	Sub-opcode	Reserved
10	1-U	1	8	4	1	2	5

Table 12-74. DJZ Control Field (C31:C0)

31–22	21	20	19–12	11–3	2–1	0
0	Control	res.	Conditional address	Reserved	Register	Int. ena.
10	1	1	8	9	2	1

Table 12-75. DJZ Data Field (D31:D0)

31–25	24–5	4–0
0	Data	Reserved
7	20	5

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.

1 DJNZ is also supported syntax. The functionality of the two instruction names is identical.

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 20-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 value != 0)
    {
    Selected register = Data field value - 1;
    Down Counter value = Data field value - 1;
    Jump to Next Program Address;
    }
else
    {
    Selected register = 00000h;
    If (Interrupt Enable == 1)
          SW interrupt flag = 1;
    Jump to conditional Address;
    }
```

12.7.3.10 ECMP (Equality Compare)

Syntax

```
ECMP {
[brk={OFF | ON}]
[next={label | 8-bit unsigned integer}]
[hr_lr={HIGH | LOW}]
[angle_comp={OFF | ON}]
[control={OFF | ON}]
[en_pin_action={OFF | ON}]
[cond_addr={label | 8-bit unsigned integer}]
pin=pin number
[action={CLEAR | SET | PULSELO | PULSEHI}]
reg={A | B | T | NONE}
[irq={OFF | ON}]
data=20-bit unsigned integer
[hr_data={5-bit unsigned integer}]
}
```

Opcode 0h, [P11:P8];

Sub-Opcode *00h, [C6:C5]*

Table 12-76. ECMP Program Field (P31:P0)

31–22	21	20	19–12	11–8	7	6	5–0
0	Res	BRK	Next program address	Opcode	hr_lr	Angle comp.	Reserved
10	1-U	1	8	4	1	1	6

Table 12-77. ECMP Control Field (C31:C0)

31–22	21	20	19–12	11–7	6–5	4–3	2–1	0
0	Control	Enable pin action	Conditional address	Pin select	Sub-opcode	Action	Register	Int. ena.
10	1	1	8	5	2	2	2	1

Table 12-78. ECMP Data Field (D31:D0)

31–25	24–5	4–0
0	Data	HR data
7	20	5

Cycles One

Register Modified Register T if selected

Description ECMP can use all pins. This instruction compares a 20-bit data value stored in the data field (D24–D5) to the value stored in the selected ALU register (A, B, or T).

If T-register is selected, and if the 20-bit data field matches, ECMP updates T-register with the 25-bit value (D24-D0).

If an HR pin is chosen (HET23-HET0) and the hr_lr bit is cleared, pin action will occur after a delay from the next resolution clock. If the hr_lr bit is set, the delay is ignored. This delay is programmed in

the data field (D4–D0). In the case of a non-HR pin, the pin action will be taken on the next loop resolution clock.

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 pin 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_flag == 1) AND (Opposite action == 1) )
            {
            If (Enable Pin Action == 1)
              {
                  Selected Pin = opposite Pin Action AT next
                  loop resolution clock;
              }
            }
          If (Interrupt Enable == 1)
            SW interrupt flag = 1;
          If (register T is selected)
            T register = Compare value (25 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
```

```
            Jump to Next Program Address;
        }
    If (Angle Comp. bit == 1 AND NAF_global == 0)
        Jump to Next Program Address;
```

12.7.3.11 ECNT (Event Count)

Syntax

```
ECNT {
[brk={OFF | ON}]
[next={label | 8-bit unsigned integer}]
[control={OFF | ON}]
[prv={OFF | ON}]
[cond_addr={label | 8-bit unsigned integer}]
pin= pin number
event={NAF | FALL | RISE |BOTH | ACCUHIGH | ACCULOW}
[reg={A | B | T | NONE}]
[irq={OFF | ON}]
data=20-bit unsigned integer
}
```

Opcode Ah;

Sub-Opcode *01h, [P6-P5]*

Table 12-79. ECNT Program Field (P31:P0)

31–22	21	20	19–12	11–8	7	6–5	4–0
0	Res	BRK	Next program address	Opcode	Res.	Sub-opcode	. Reserved
10	1-U	1	8	4	1	2	5

Table 12-80. ECNT Control Field (C31:C0)

31–22	21	20	19–12	11–7	6–4	3	2–1	0
0	Control	Prv.	Conditional address	Pin select	Event	Res.	Register	Int. en.
10	1	1	8	5	3	1	2	1

Table 12-81. ECNT Data Field (D31:D0)

31–25	24–5	4–0
0	Data	Reserved
7	20	5

Cycles	One cycle
Register Modified	None
Description	This instruction defines a specialized 20-bit virtual counter used as an event counter or pulse accumulator (see Table 12-82). The counter value is stored in the data field [D24:D5]. When an event count condition is specified, the counter value is incremented on a pin edge condition or on the NAF_global condition (NAF_global is defined in ACNT). An edge detect is performed by comparing the current pin level, sampled at loop start, to the previous pin level stored in Prv. bit. The ECNT instruction updates the Prv. bit each time an edge is detected regardless of whether the edge is selected. This instruction can be used with all pins.

event — The event that triggers the counter.

Table 12-82. 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 — 20-bit integer value serving as a counter.
Default: 0.

Execution

```
If (Event count condition is true according to bits [C6:C4] (see table 1-17))
{
    Selected register = Immediate Data Field + 1;
    Immediate Data Field = Immediate Data Field + 1;
    If (Interrupt Enable == 1)
          SW interrupt flag = 1;
    Jump to Conditional Address;
    }
else
    Jump to Next Program Address;
Prv. bit = Selected Pin level; (Always executed)
```

12.7.3.12 MCMP (Magnitude Compare)

Syntax

```
MCMP {
[brk={OFF | ON}]
[next={label | 8-bit unsigned integer}]
[hr_lr={LOW |HIGH}]
[angle_comp={OFF | ON}]
[savesub={OFF | ON}]
[control={OFF | ON}]
[en_pin_action={OFF | ON}]
[cond_addr={label | 4-bit unsigned integer}]
pin= pin number
order={REG_GE_DATA | DATA_GE_REG}
[action={CLEAR | SET | PULSELO | PULSEHI}]
reg={A | B | T | NONE}
[irq={OFF | ON}]
data=20-bit unsigned integer
[hr_data=5-bit unsigned integer]
}
```

Opcode 0h, [P11–P8];

Sub-Opcode *1h, C6*

Table 12-83. MCMP Program Field (P31:P0)

31–22	21	20	19–12	11–8	7	6	5	4	3–0
0	Res	BRK	Next program address	Opcode	hr_lr	Angle comp.	res.	Save sub.	res.
10	1-U	1	8	4	1	1	1	1	4

Table 12-84. MCMP Control Field (C31:C0)

31–22	21	20	19–12	11–7	6	5	4–3	2–1	0
0	Control	Enable pin action	Conditional address	Pin select	Sub-op code	Order	Action	Register	Int. ena.
10	1	1	8	5	1	1	2	2	1

Table 12-85. MCMP Data Field (D31:D0)

31–25	24–5	4–0
0	Data	HR data
7	20	5

Cycles One

Register Modified Register T (implicitly)

Description This instruction compares the magnitude of the 20-bit data value stored in the data field (D24-D5) and the 20-bit value stored in the selected ALU register (A, B, or T). The register select coding for C2 and C1 are shown in Table 12-40.

Note: The Difference Between Compare Values
The difference between the two data **values must not exceed** (2^{19}) - 1.

If an HR pin is chosen (HET23-HET0) and the hr_lr bit is reset, pin action will occur after a delay from the next resolution clock. If the hr_lr bit is set, the delay is ignored. This delay is programmed in the

data field (D4-D0). In the case of a non-HR pin, the pin action will be taken on the next resolution clock.

When the data value matches, an output pin can be set or reset according to the pin action bit (C4). The pin will not change states if the enable pin action bit (C20) is reset.

MCMP uses the zero flag set to generate opposite pin action (synchronized to the loop resolution clock). The save sub bit (P4) provides the option to save the result of a subtraction into register T.

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 20 bits).
Default: OFF.

order — Specifies the order of the operands for the comparison.

Table 12-86. 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 register and data field values are equivalent. 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 pin display. The default value for an unspecified bit is 0.

Execution

```
If (Angle Comp. bit == 0 OR (Angle Comp. bit == 1 AND NAF_global ==
1))
    {
    If ((C5 == 1 AND (Immediate data value - Selected register) ≥0) OR
    (C5 == 0 AND (Selected register - Immediate data value) ≥0))
          {
          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_flag == 1) AND (Opposite action == 1) )
            {
            If (Enable Pin Action == 1)
              {
                 Selected Pin = opposite Pin Action AT next
                 loop resolution clock;
              }
```

```
                    }
                If (Interrupt Enable == 1)
                    SW interrupt flag = 1;
                If (Save sub. == 1)
                    {
                    If ([C5] == 1)
                        T register = Immediate data value - Selected register;
                    If ([C5] == 0)
                        T register = Selected register - Immediate data value;
                    }
                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
                Jump to Next Program Address;
            }
        If (Angle Comp. bit == 1 AND NAF_global == 0)
            Jump to Next Program Address;
```

12.7.3.13 MOV32 (Data Move 32)

Syntax

```
MOV32 {
[brk={OFF | ON}]
[next={label | 8-bit unsigned integer}]
remote={label | 8-bit unsigned integer}
[control={OFF | ON}]
[init={OFF | ON}]
type={IMTOREG | IMTOREG&REM | REGTOREM | REMTOREG}
reg={A | B | T | NONE}
[data=20-bit unsigned integer]
[hr_data=5-bit unsigned integer]
}
```

Opcode 4h, [P11:P8];

Sub-Opcode *C5=0*

Table 12-87. MOV32 Program Field (P31:P0)

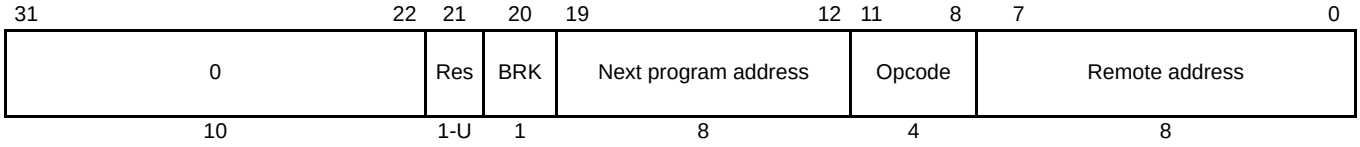


Table 12-88. MOV32 Control Field (C31:C0)

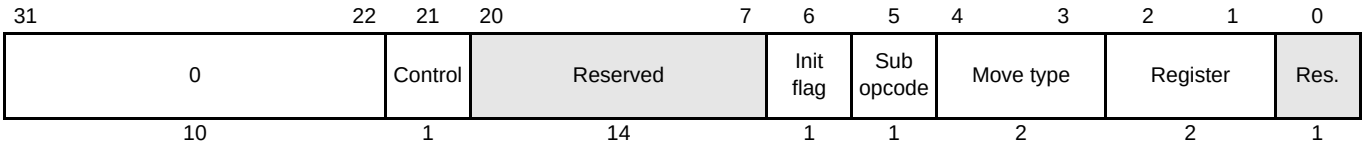
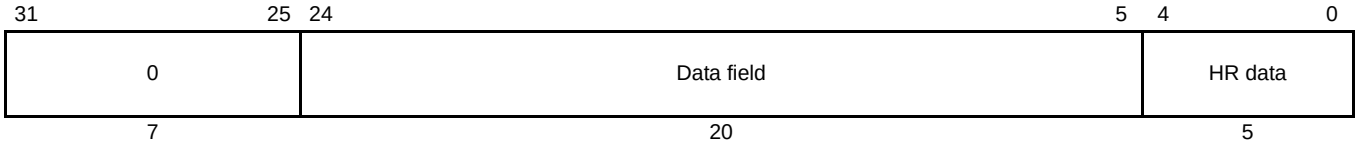


Table 12-89. MOV32 Data Field (D31:D0)

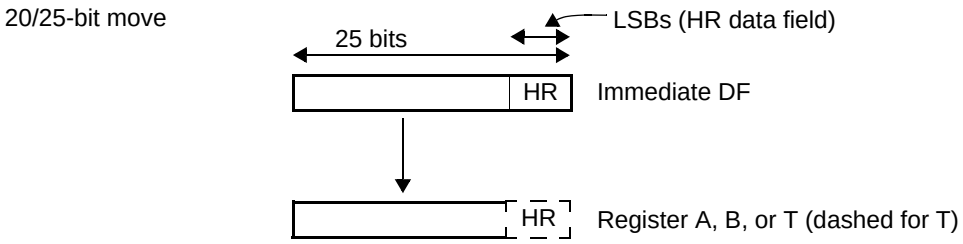


Cycles	One or two cycles
Register Modified	Selected register (A, B or T)
Description	MOV32 replaces the selected ALU register and/or the data field values at the remote address location depending on the move type. Table 12-40 shows the C2 and C1 bit encoding for selecting the desired register. Table 12-90 shows the [C4:C3] bit encoding for selecting the destination for a data move. Figure 12-68 through Figure 12-71 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.
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 12-90. Move Type Encoding Selection

Move Type	C4	C3	Source	Destination(s)	Cycles
IMTOREG	0	0	Immediate data field	Register A, B, or T	1
IMTOREG&REM	0	1	Immediate data field	Remote data field & register A, B, or T	1
REGTOREM	1	0	Register A, B, or T	Remote data field	1
REMTOREG	1	1	Remote data field	Register A, B, or T	2

Figure 12-68. 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 assembler 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 20-bit integer value to be written to the remote data field or selected register.

hr_data — (Optional) HR pin delay. The default value for an unspecified bit is 0.

Figure 12-69. MOV32 Move Operation for IMTOREG&REM (Case 01)

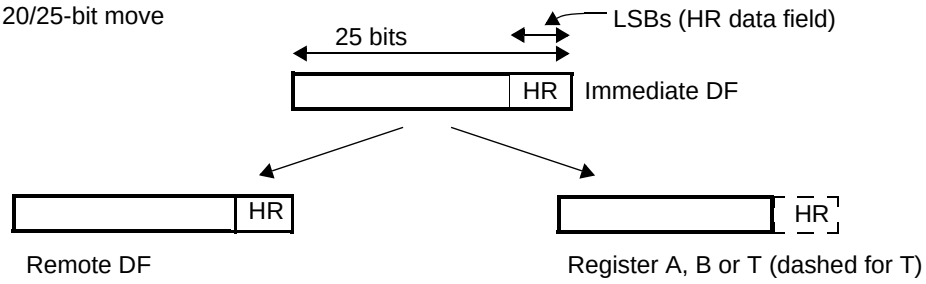


Figure 12-70. MOV32 Move Operation for REGTOREM (Case 10)

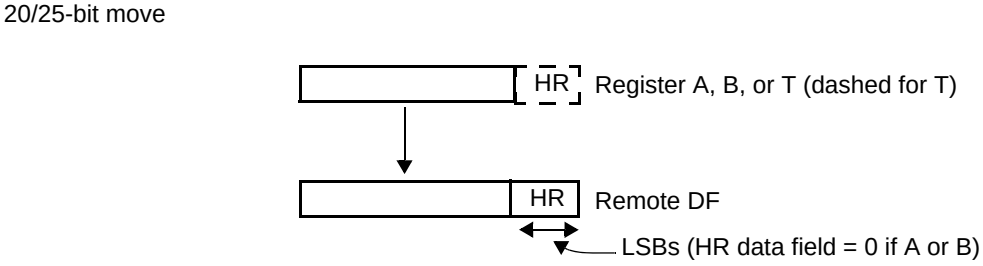
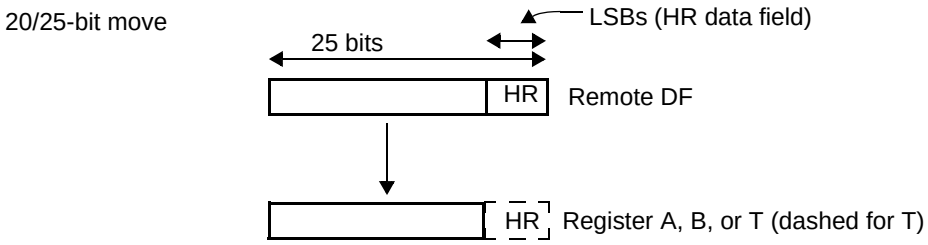


Figure 12-71. MOV32 Move Operation for REMTOREG (Case 11)



Execution

```
switch (C4:C3)
    {
    case 00:
          Selected register = Immediate Data Field;
    case 01:
          Selected register = Immediate Data Field;
          Remote Data Field = Immediate Data Field;
    case 10:
          Remote Data Field = Selected register;
    case 11:
          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;
```

12.7.3.14 MOV64 (Data Move 64)

Syntax

```
MOV64 {
[brk={OFF | ON}]
[next={label | 8-bit unsigned integer}]
remote={label | 8-bit unsigned integer}]
[control={OFF | ON}]
[en_pin_action={OFF | ON}]
[cond_addr={label | 4-bit unsigned integer}]
[pin= pin number]
comp_mode={ECMP | SCMP | MCMP | ACMP}
[action={CLEAR | SET | PULSELO | PULSEHI}]
reg={A | B | T | NONE}
[irq={OFF | ON}]
[data=20-bit unsigned integer]
[hr_data=5-bit unsigned integer]
}
```

- or -

Syntax

```
MOV64 {
[brk={OFF | ON}]
[next={label | 8-bit unsigned integer}]
remote={label | 8-bit unsigned integer}]
[control={OFF | ON}]
[cntl_val=21-bit unsigned integer]
[data=20-bit unsigned integer]
[hr_data=5-bit unsigned integer]

}
```

Opcode 1h, [P11:P8]

Table 12-91. MOV64 Program Field (P31:P0)

31–22	21	20	19–12	11–8	7–0
0	Res	BRK	Next program address	Opcode	Remote address
10	1-U	1	8	4	8

Table 12-92. MOV64 Control Field (C31:C0)

31–22	21	20	19–12	11–7	6	5	4–3	2–1	0
0	Control	Enable pin action	Conditional address	Pin select	Comp. mode		Action	Register	Int. ena.
10	1	1	8	5	2		2	2	1

Table 12-93. MOV64 Data Field (D31:D0)

31–25	24–5	4–0
0	Data	HR data
7	20	5

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 21-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 12-72.

Figure 12-72. MOV64 Move Operation

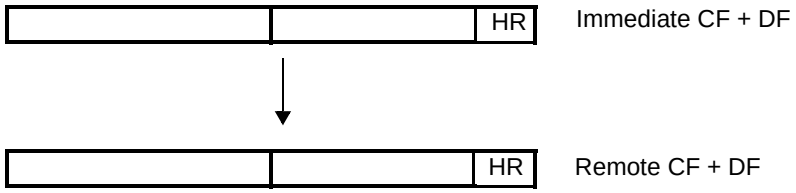


Table 12-94. MOV64 Control Field Descriptions

Control Field	Description
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 20-bit initial count value for the data field. If omitted, the field defaults to 0.
hr_data	(Optional) HR pin delay. The default value for an unspecified bit is 0.
comp_mode	Selects the comparison mode type to be used.

Table 12-95. Comparison Type Encoding Format

comp_mode	C6	C5
ECMP	0	0
SCMP	0	1
MCMP	1	0
ACMP	1	1

Execution

```
Remote Data Field = Immediate Data Field;
Remote Control Field = Immediate control Field;
Jump to Next Program Address;
```

12.7.3.15 PCNT (Period/Pulse Count)

Syntax

```
PCNT {
[brk={OFF | ON}]
[next={label | 8-bit unsigned integer}]
[irq={OFF | ON}]
type={RISE2FALL | FALL2RISE | FALL2FALL | RISE2RISE}
pin=pin number
[control={OFF | ON}]
[prv={OFF | ON}]
[period=20-bit unsigned integer]
[data=20-bit unsigned integer]
[hr_data=5-bit unsigned integer]
}
```

Opcode 7h, [P11:P8]

Table 12-96. PCNT Program Field (P31:P0)

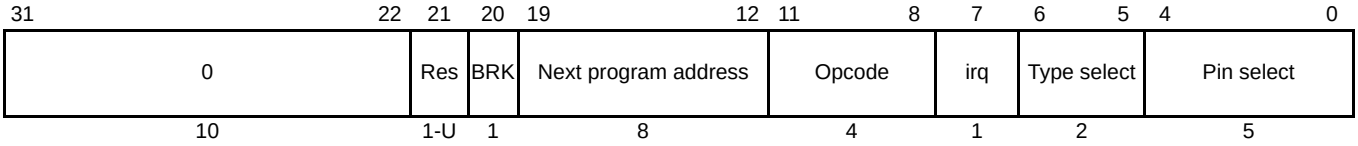


Table 12-97. PCNT Control Field (C31:C0)

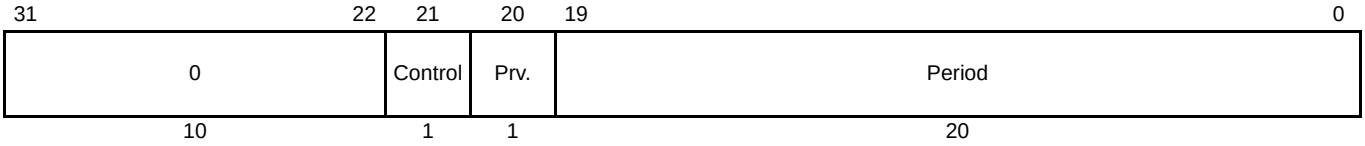
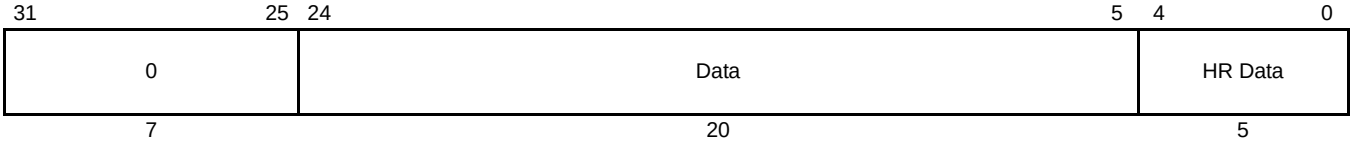


Table 12-98. PCNT Data Field (D31:D0)



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 [C19:C0] and in the register A is incremented on each timer resolution. If an HR pin is selected (HET23-HET0), then PCNT will use the HR structure on the pin to measure an HR period/pulse count value.

irq (Optional) 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.

type (Optional) Determines the type of counter that is implemented.

Table 12-99. Counter Type Encoding Format

	P6	P5	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

	P6	P5	Period/Pulse Select	Reset On	Capture On
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 20-bit integer value that holds the previous counter value. The previous counter value is also stored in register A.
Default: 0.

data — 20-bit integer value serving as a counter.
Default: 0.

hr_data — HR pin delay.
Default: 0.

If the control bit is set, the period value is automatically cleared after the main CPU performs a 64-bit read access.

If ***period-measure*** is selected, PCNT captures the counter value into the period/pulse data field [D24:D5] on the selected edge. The HR structure provides HR capture field [D4:D0]. The counter value [C19:C0] is reset on the same edge. The period value is a 25-bit value.

If ***pulse-measure*** is selected, PCNT captures the counter value into the period/pulse count field [D24:D5] on the selected edge. The HR structure provides HR capture field [D4:D0]. The counter value [C19:C0] is reset on the next opposite edge. The pulse value is a 25-bit value.

An edge detect is performed by comparing the current pin level, sampled at loop start, to the previous pin level stored in Prv. bit.

When the overflow count (all 1's in the counter value) is reached, PCNT stops counting until the next reset edge is detected. Table 12-39 shows the pin encoding for the pin select field [P4:P0].

Execution

```
Z = 0;
If (Period/Pulse select [P6,P5] == 1X)
    {
    If (Period value != FFFFFh)
          {
          Register A = Period value + 1;
          Period value = Period value + 1;
          }
    elseIf (specified edge not detected on selected pin)
          Register A = FFFFFh;
          HR capture value = 1Fh;
    If (specified edge detected on selected pin)
          {
          Z = 1;
          If (Period value == FFFFFh)
            {
            Register A = FFFFFh;
            Data field = FFFFFh;
            }
          else
            {
            Data field = Period value + 1;
            HR capture value = selected HR counter;
            Period value = 00000h;
```

```
                If (Interrupt Enable == 1)
                  SW interrupt flag = 1;
                Jump to Next Program Address;
                }
              }
    }
else /*** Pulse mode ***/
    {
    If (Period value != FFFFFh)
          {
          Register A = Period value + 1;
          Period value = Period value + 1;
          }
    If (specified reset edge is detected on selected pin)
          Period value = 00000h;
    If (specified capture edge is detected on selected pin)
          {
          Z = 1;
          If (Period value == FFFFFh)
            {
            Register A = FFFFFh;
            Data field = FFFFFh;
            HR capture value = 11111;
            }
          else
            {
            Data field = Period value + 1;
            HR Capture Value = selected HR counter value;
            If (Interrupt Enable == 1)
              SW interrupt flag = 1;
            }
          }
    }
    Jump to Next Program Address;
Prv. bit = Selected Pin level;
```

12.7.3.16 PWCNT (Pulse Width Count)

Syntax

```
PWCNT {
[brk={OFF | ON}]
[next={label | 8-bit unsigned integer}]
[hr_lr={HIGH | LOW}]
[control={OFF | ON}]
[cond_addr={label | 8-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=20-bit unsigned integer
[hr_data=5-bit unsigned integer]
}
```

Opcode — Ah, [P11:P8];

Sub-Opcode — *11h, [P6-P5]*

Table 12-100. PWCNT Program Field (P31:P0)

31–22	21	20	19–12	11–8	7	6–5	4–0
0	Res	BRK	Next program address	Opcode	hr_lr	Sub-opcode	Reserved
10	1-U	1	8	4	1	2	5

Table 12-101. PWCNT Control Field (C31:C0)

31–22	21	20	19–12	11–7	6–5	4–3	2–1	0
0	Control	En. pin action	Conditional address	Pin select	Reserved	Action	Register	Int. enab.
10	1	1	8	5	2	1	1	1

Table 12-102. PWCNT Data Field (D31:D0)

31–25	24–5	4–0
0	Data	HR data
7	20	5

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 an HR pin is chosen (HET23–HET0) and the hr_lr bit is reset, the opposite pin action will be taken after a delay from the next resolution clock. If the hr_lr bit is set, the delay is ignored. This delay is

programmed in bits [D4:D0]. In the case of a non-HR pin, the opposite pin action is taken on the next resolution clock.

irq — ON generates an interrupt when the data field value reaches 0. No interrupt is generated for OFF
Default: OFF.

data — 20-bit integer value serving as a counter.

hr_data — HR pin 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 = 00000h;
    Data field value = 00000h;
    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)
    SW interrupt flag = 1;
    Jump to Conditional Address
    }
```

12.7.3.17 RADM64 (Register Add Move)

Syntax

```
RADM64 {
[brk={OFF | ON}]
[next={label | 8-bit unsigned integer}]
remote={label | 8-bit unsigned integer}
[control={OFF | ON}]
[en_pin_action={OFF | ON}]
[cond_addr={label | 4-bit unsigned integer}]
[pin= pin number]
comp_mode={ECMP | SCMP | MCMP | ACMP}
[action={CLEAR | SET | PULSELO | PULSEHI}]
reg={A | B | T | NONE}
[irq={OFF | ON}]
data=20-bit unsigned integer
[hr_data=5-bit unsigned integer]
}
```

or

Syntax

```
RADM64 {
[brk={OFF | ON}]
[next={label | 8-bit unsigned integer}]
remote={label | 8-bit unsigned integer}
[cntl_val = 21-bit unsigned integer]
[control={OFF | ON}]
data=20-bit unsigned integer
[hr_data=5-bit unsigned integer]
}
```

Opcode 3h, [P11:P8]

Table 12-103. RADM64 Program Field (P31:P0)

31–22	21	20	19–12	11–8	7–0
0	Res	BRK	Next program address	Opcode	Remote
10	1-U	1	8	4	8

Table 12-104. RADM64 Control Field (C31:C0)

31–22	21	20	19–12	11–7	6–5	4–3	2–1	0
0	Control	Enable pin action	Conditional address	Pin select	Comp. mode	Action	Register	Int. ena.
10	1	1	8	5	2	2	2	1

Table 12-105. RADM64 Data Field (D31:D0)

31–25	24–5	4–0
0	Data	HR data
7	20	5

Cycles One

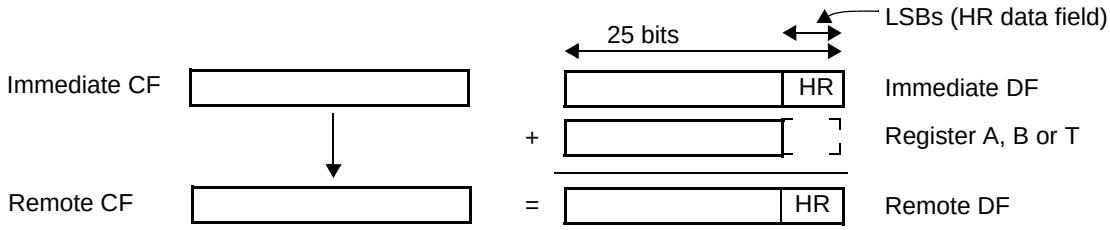
Register Modified Selected register (A, B or T)

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 of the T-register selected,

the addition is a 25-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 21-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 12-73.

Figure 12-73. RADM64 Add and Move Operation



comp_mode Selects the comparison mode type to be used.

Table 12-106. Comparison Type Encoding Format

comp_mode	C6	C5
ECMP	0	0
SCMP	0	1
MCMP	1	0
ACMP	1	1

Table 12-107. Control Field Description

Control Field	Description
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 20-bit initial value for the data field. If omitted, the field defaults to 0.
hr_data	Five least significant bits of the 25-bit data field. Default: 0.
cntl_val	Specifies the 21 least significant bits of the Control field.

Execution

```
Remote Data Field = Selected register + Immediate Data Field
(including HR data field);

Remote Control Field = Immediate Control Field;

Jump to Next Program Address;
```

12.7.3.18 SCMP (Sequence Compare)

Syntax

SCMP {
[brk={OFF | ON}]
[next=*{label | 8-bit unsigned integer}]*
[control={OFF | ON}]
[en_pin_action={OFF | ON}]
cond_addr=*{label | 8-bit unsigned integer}*
pin=pin number
[action={CLEAR | SET}]
[restart={OFF | ON}]
[irq={OFF | ON}]
data=*20-bit unsigned integer*
}

Opcode 0h, [P11–P8];

Sub-Opcode [C6-C5]=01

Table 12-108. SCMP Program Field (P31:P0)

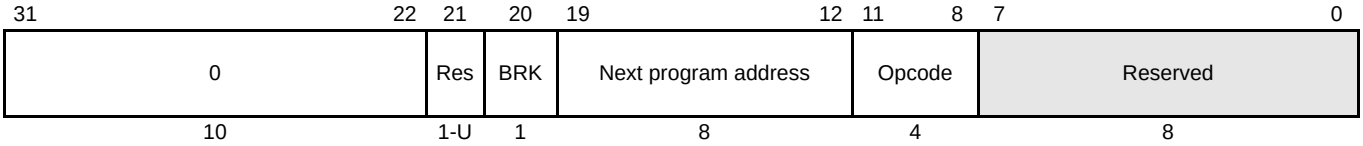


Table 12-109. SCMP Control Field (C31:C0)

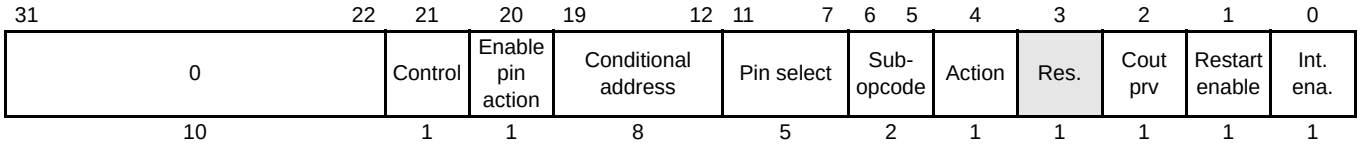
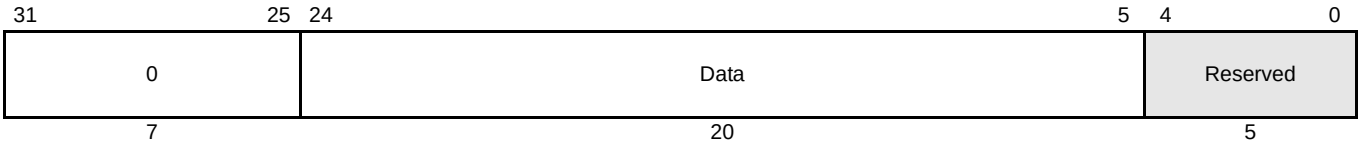


Table 12-110. SCMP Data Field (D31:D0)



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 (C12) 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 (C20) 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 ECMP 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 register and data field values are equivalent. No interrupt is generated when the field is OFF.
Default: OFF.

data — Specifies the 20-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)
    {
    C12 = 1;
    Immediate Data Field = Register A;
    Jump to Next Program Address;
    }
If (Angle Mode is selected (C12 == 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)
            SW interrupt flag = 1;
          Immediate Data Field = Register A;
          C12 = 1; /*** switch to Time Mode ***/
          Jump to Conditional Address;
          }

    else
          Jump to Next Program Address;
    }
If (Time Mode is selected (C12 == 1)) AND ((Restart En. == 1 AND X ==
0) OR Restart En. == 0)
    {
```

```
    Register T = Register A - Immediate Data Field;
    (Result of subtract must not exceed (2^19) - 1)
    Jump to Conditional Program Address;
    }
Cout_prv = Cout; (always executed)
```

Example 1. SCMP Example

```
  ; Time Counter in register A
A0CNT{reg=A, max=0x0fffff}
  ; Angle Counter (320 degree) in register B
A1CNT{reg=B, max=0x3C00, irq=OFF, angle_count=ON, data=0x400}
  ; Angle compare to set pulse on CC23 when match X flag is set
A2ACMP{next=A3, en_pin_action=ON, cond_addr=A3, pin=CC23, ac-
tion=SET, reg=B, irq=OFF, data=0x0220}
  ; Transfer match angle into time, when X is set transfer A into DF,
switch in time mode and jump to MCMP else next ACMP (next loop reso-
lution)
A3SCMP{next=A6, en_pin_action=ON, cond_addr=A5, pin=CC23, ac-
tion=SET, restart=ON, irq=OFF, data=0x0230}
  ; restart the sequence compare and jump to next resolution
A4MOV64{next=A6, remote=A3, en_pin_action=ON, cond_addr=A4, pin=CC23,
comp_mode=SCMP, action=SET, reg=B, data=0x0230}
  ; When T (= Reg. A - SCMP Data field) > compare value then reset
the pin and jump to MOV64 to re-initialize the SCMP Control and Data
field.
A5MCMP{next=A6, en_pin_action=ON, pin=CC23, hr_lr=LOW, action=CLEAR,
order=REG_GE_DATA, cond_addr=A4, reg=T, data=0x214, hr_data=0x10}
  ; Branch to first instruction (end of the loop)
A6BR{next=A0, cond_addr=A0, pin=CC25, event=NOCOND}
```

This program generates a pulse always at the same position of a wheel (ACMP instruction) and the pulse lasts for a certain time (value in MCMP instruction) whatever the speed of the wheel.

12.7.3.19 SCNT (Step Count)

Syntax

```
SCNT {
[brk={OFF | ON}]
[next={label | 8-bit unsigned integer}]
step={8 | 16 | 32 | 64}
[control={OFF | ON}]
gapstart=20-bit unsigned integer
[data=20-bit unsigned integer]
}
```

Opcode: Ah, [P11:P8];

Sub-Opcode: *[P6-P5]=00*

Table 12-111. SCNT Program Field (P31:P0)

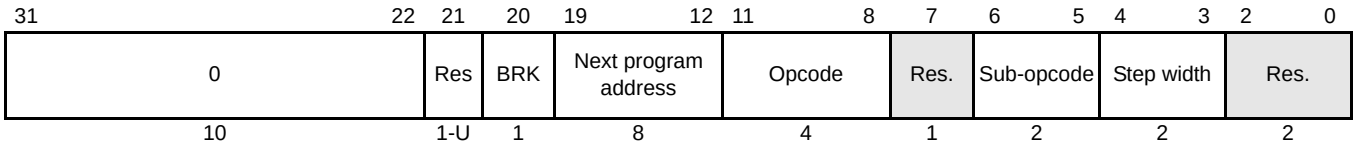


Table 12-112. SCNT Control Field (C31:C0)

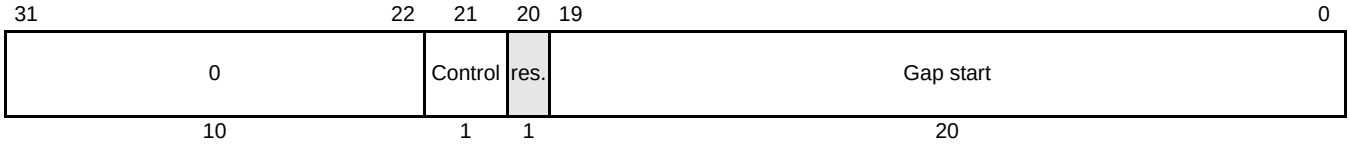
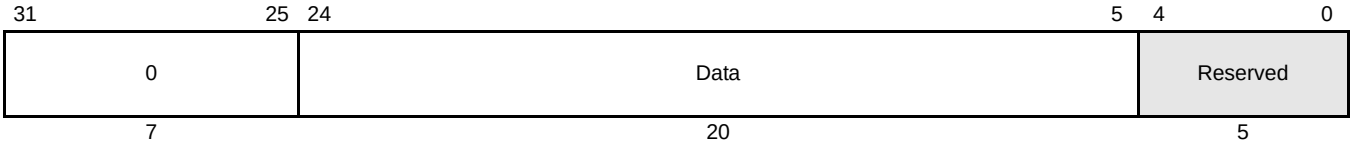


Table 12-113. SCNT Data Field (D31:D0)



Cycles: Three

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, [P4:P3]. The bit encoding for this field is defined in Table 12-114:

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 12-114.

Table 12-114. Step Width Encoding for SCNT

P4	P3	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 x (actual teeth on gear - 1)) + 1.

data — Specifies the 20-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 = P4;
SWF0 = P3;
Z = 0;
If (register T == 00000h)
    Jump to Next Program Address;
else
    {
    If (DCF == 1 OR ACF == 1)
          {
          Data Field register = 00000h;
          Counter value = 00000h;
          }
    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;
```

12.7.3.20 SHFT (Shift)

Syntax

```
SHFT {
[brk={OFF | ON}]
[next={label | 8-bit unsigned integer}]
smode={OR0 | OL0 | OR1 | OL1 | ORZ | OLZ | IRM | ILL | IRZ | ILZ}
[control={OFF | ON}]
[prv={OFF | ON}]
[cond_addr={label | 8-bit unsigned integer}]
cond={UNC | FALL | RISE}
pin=pin number
[reg={A | B | T | NONE}]
[irq={OFF | ON}]
data={20-bit unsigned integer}
}
```

Opcode Fh, [P11:P8]

Table 12-115. SHFT Program Field (P31:P0)

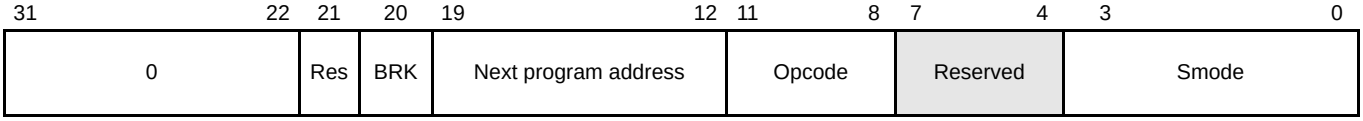


Table 12-116. SHFT Control Field (C31:C0)

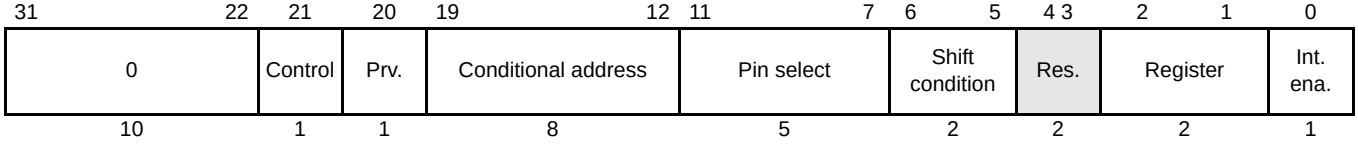
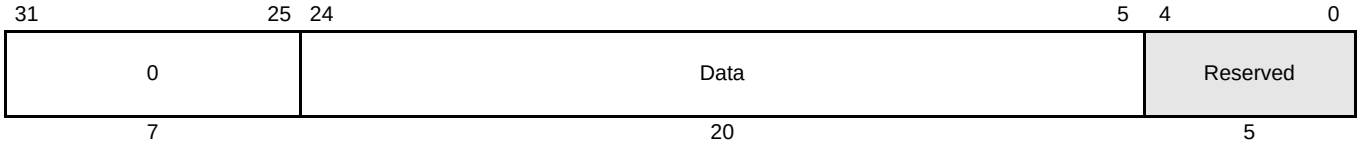


Table 12-117. SHFT Data Field (D31:D0)



Cycles One

Register Modified Selected register (A, B or T)

Description This instruction **may only be used on pins CC24-CC31.**

This instruction shift the data field of the Instruction. HET 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 HET31 or shift always), register for data storage (A, B, or T), and the data pin.

smode Shift mode.

Table 12-118. 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

smode	P3	P2	P1	P0	Operation	
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 12-119. SHIFT Condition Encoding

C6	C5	Shift Condition
0	X	Always
1	0	Rising edge of HET31
1	1	Falling edge of HET31

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 20-bit value for the data field.

Execution

```
If (SHIFT condition == 0X)
OR (SHIFT condition == 10 AND CC31 rising edge)
OR (SHIFT condition == 11 AND CC31 falling edge)
    {
    Shift Immediate Data Field by one bit according to bits [P3:P0];
    (see Table 12-118)
    Immediate Data Field = Result of the shift;
    Selected register = Result of the shift;

    If ((Immediate Data Field == all 0's AND [P3:P0] == 000X) OR (Im-
    mediate Data Field == all 1's AND [P3:P0] == 001X))
          {
          Z = 1;
          }
    else if ([P3:P0] == 1010)
          {
          Z = LSB of the Immediate Data Field;
          }
    else if ([P3:P0] == 1011)
          {
          Z = MSB of the Immediate Data Field;
          }
    else
          {
          Z = 0;
          }

    If ((Immediate Data Field == all 0's OR
            (Immediate Data Field == all 1's)
            AND (Interrupt Enable == 1))
          {
```

```
            SW interrupt flag = 1;
            Jump to Conditional Address;
            }
      else
            {
            Jump to Next Program Address;
            }
      }
   Prv. bit = CC31 Pin level; (Always executed)
   Jump to Next Program Address;
```

Note:
The immediate data field evaluates all 0s or all 1s and is performed before the shift operation.

12.7.3.21 WCAP (Software Capture Word)

Syntax

```
WCAP {
[brk={OFF | ON}]
[next={label | 8-bit unsigned integer}]
[hr_lr={HIGH | LOW}]
[control={OFF | ON}]
[prv={OFF | ON}]
[cond_addr={label | 8-bit unsigned integer}]
pin=pin number
event={NOCOND | FALL | RISE | BOTH}
reg ={A | B | T | NONE}
[irq={OFF | ON}]
data=20-bit unsigned integer
[hr_data=5-bit unsigned integer]
}
```

Opcode — Bh, [P11:P8]

Table 12-120. WCAP Program Field (P31:P0)

31–22	21	20	19–12	11–8	7	6–0
0	Res	BRK	Next program address	Opcode	hr_lr	Reserved
10	1-U	1	8	4	1	7

Table 12-121. WCAP Control Field (C31:C0)

31–22	21	20	19–12	11–7	6–5	4–3	2–1	0
0	Control	Prv.	Conditional address	Pin select	Capture condition	Reserved	Register	Int. ena.
10	1	1	8	5	2	2	2	1

Table 12-122. WCAP Data Field (D31:D0)

31–25	24–5	4–0
0	Data	HR data
7	20	5

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 an HR pin is selected (HET23-HET0) and 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. Edge detection is performed by comparing the current pin level sampled at loop start to the previous pin level stored in the Prv. (Previous) bit [C20].

This instruction updates the Prv. bit each time an edge is detected regardless of whether the edge is selected.

event — Specifies the event that triggers the capture.

Table 12-123. 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 edges

irq — ON generates an interrupt when the capture condition is met. No interrupt is generated for OFF
Default: OFF.

data — Specifies the 20-bit integer value to be written to the data field or selected register.

hr_data — HR capture value.
Default: 0.

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)
          SW interrupt flag = 1;
    Jump to Conditional Address;
    }
else
    Jump to Next Program Address;
Prv. bit = Selected Pin level;
```

Cortex M3 LOCKUP Reset Module (LRM)

This chapter describes the LOCKUP reset module (LRM) used with the ARM Cortex M3 CPU in the TMS470M Series architecture.

Topic **Page**

13.1 Overview 716
13.2 Features 716

13.1 Overview

The ARM Cortex M3 processor has a unique failure state known as LOCKUP. In order to consistently handle this operating state, it is necessary to use a standard module that manages the handling of Cortex M3 while in LOCKUP state. For more details of how LOCKUP is generated please refer to the ARM documentation. This document describes only the LRM logic which is used to handle the LOCKUP state in TMS470M Series Cortex M3 devices.

13.2 Features

The LOCKUP Reset Module (LRM) is a simple module with two functions. The LRM translates the LOCKUP signal from the Cortex M3 from level to pulse on rising edge detection so that the resulting signal may be used as an ESM input. In addition, the LRM provides a small watchdog timer which may generate a reset in case the LOCKUP condition cannot be cleared by softwareThe reset request is connected to system module to generate a warm system reset.

13.2.1 LOCKUP Pulse Generation

The LOCKUP signal from Cortex M3 is effectively a level interrupt. It is desirable to input this signal to the ESM block. The ESM only supports pulse inputs. The LRM takes the input LOCKUP signal and generates a pulse version to be connected to the ESM.

The LRM generates a low/high/low pulse of one VCLK period in duration, at any time when a rising edge is detected on LOCKUP.

13.2.2 LOCKUP Watchdog Reset

It may be possible in some cases to recover the M3 CPU from the LOCKUP state via NMI handler software. The LRM provides a mechanism to put the M3 into a usable state in case the software recovery method fails. For this purpose, a small watchdog timer is implemented in LRM. An 8 bit countdown timer is implemented. The maximum value will be set in the counter after warm reset. The reset value is 0xFF. So the reset is generated after 256 VCLK cycles.

Count down will start when a rising edge is detected on the LOCKUP signal. When the counter reaches zero, the LRM checks the current state of the LOCKUP signal. If LOCKUP remains high, the LRM will generate a reset signal. This reset implementation is device specific On this device the reset request is connected to system module to generate a warm system reset.

The LRM counter value will be reset when the counter reaches zero or on a warm system reset or falling edge detected on LOCKUP. The counter reset on falling edge detect ensures that the core or warm system reset will not be triggered if software was able to successfully clear the LOCKUP state.

CPU Self Test Controller (LBIST) Module

This chapter describes of the Logic BIST concept and the CPU Self Test Controller implemented in the TMS470M Series of microcontrollers

Topic **Page**

14.1 General Description 718
14.2 Deterministic Logic BIST concept. 719
14.3 STC Block diagram 721
14.4 Module Description. 723
14.5 Application Self Test Flow Chart. 724
14.6 SelfTest Execution Flow. 725
14.7 Self Test Completion and Error Generation 726
14.8 STC clock Divider 727
14.9 Control Registers 728
14.10ROM Organization 739
14.11Timing Diagrams 741

14.1 General Description

The CPU Self Test Controller (STC) is used to test the ARM CPU core using the Deterministic Logic BIST (LBIST) Controller as the test engine.

Software based Selftest program for the cores are available but offers slightly less test coverage. Also due to the complexity of the soft cores it is very difficult to achieve the required coverage and also the program size will be much larger. For these complex cores, on-chip logic BIST support for the self test is the preferred solution.

The main features of this solution includes:

1. Implement Logic BIST controller along with On-chip Self-Test controller for the synthesizable CPU cores which enables to achieve high test coverage
2. Ability to divide the complete test run into independent test sets (intervals).
3. Capable of running the complete test as well as running few intervals at a time.
4. Ability to continue from the last executed interval (test set) as well as ability to restart from the beginning (First test set).
5. Complete isolation of the self tested CPU core with rest of the system during the self-test run.
   a. The Self tested CPU core master bus transaction signals are configured to be in Idle mode during the self test run.
   b. Any master access to the CPU Core under Self Test will be held till the completion of the self test.
6. Ability to capture the Failure interval number.
7. Timeout counter for the CPU self test run as a fail-safe feature.
8. Able to read the MISR data (shifted from LBIST controller) of the last executed interval of the selftest run for debugging purposes
9. Supports single CPU as well as dual CPU architecture with two LBIST controllers.
10. Generically parameterizable options for the number of shadow scan channels to support different LBIST controller configurations with different shadow scan chains.
11. Pre-scalar implemented on STC clock enables running LBIST on core with reduced current consumption

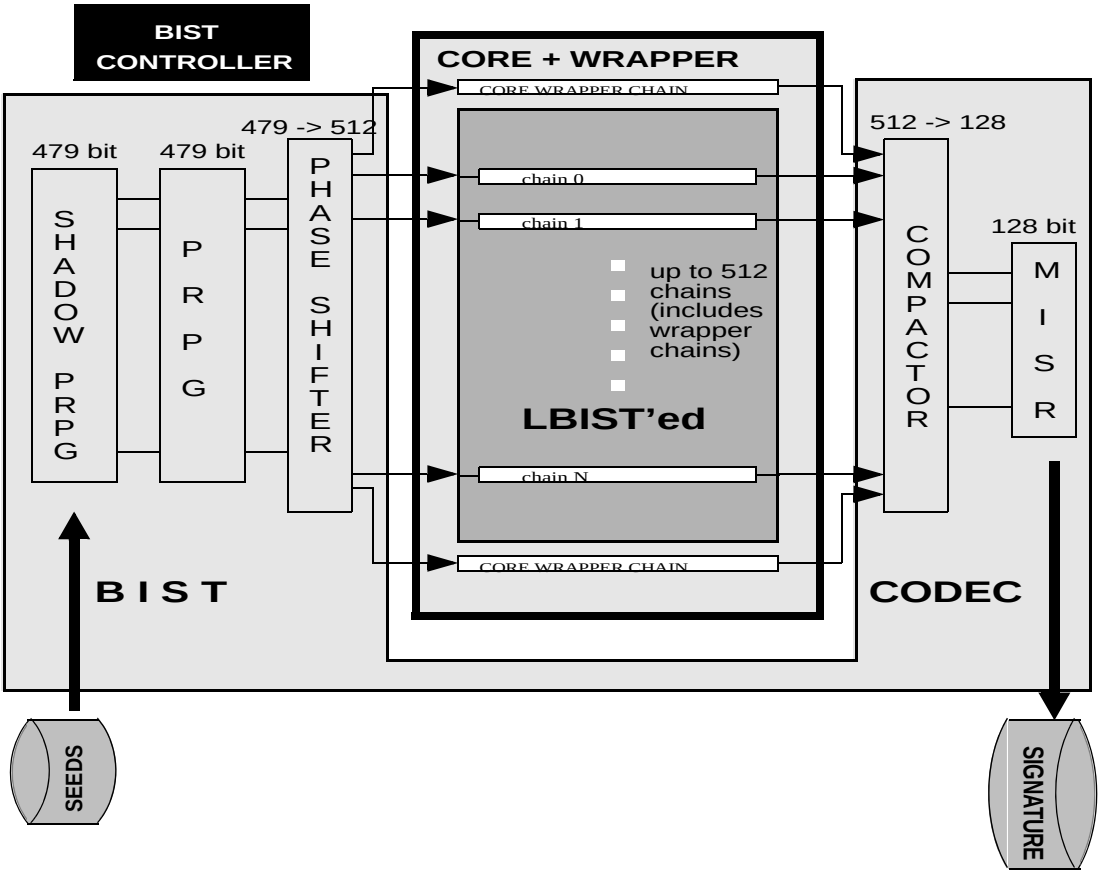
Note: There are minor differences in logic BIST (LBIST) and Deterministic Logic BIST (DBIST) implementations and how different tool vendors name it. This document refers to the DBIST implementation. But the terms LBIST and DBIST are used interchangeably.

14.2 Deterministic Logic BIST concept

Deterministic Logic BIST is a methodology which moves the test pattern generation to on-chip. Logic BIST will be implemented on functional partitions (BISTe'd CORES) that are speed critical and have high gate count. A conceptual diagram of Deterministic Logic BIST implementation is shown in figure below. The architecture contains a Linear feedback shift register (LFSR) which is initialized to a particular value or seeds. The DBIST tools uses deterministic test patterns to produce the seeds or initial values. This functions as a pseudo-random pattern generator (PRPG). The test patterns are compressed in PRPG seeds which are applied through the PRPG shadow registers. The patterns pass through a phase shifter (LFSR) which basically converts the patterns generated by the PRPG into a 2-dimensional arrays of parallel scan chains which are scanned into the design under test (BISTe'd CORE). The idea is to have as many parallel scan chains as possible which are kept short which increases the controllability and observability of the design with the minimum set of patterns. The Deterministic Logic BIST tool supports up to 512 parallel scan chains. The captured data from the LBIST core are loaded in parallel into signature analyzer which compacts the scan outputs into a signature.

The signature analyzer is a modified LFSR know as multiple input signature register (MISR), A compactor is a combinational logic block which reduces the scan chain outputs coming from the Logic BIST core so that the MISR can be smaller. The shadow PRPG, PRPG, Phase shifter, compactor and MISR together constitute an LBIST CODEC. THE LBIST CODEC in turn is controlled by an LBIST controller. A wrapper is also generated around the BIST'ed CORE this contains all the Input bounding cells suppress X propagation to the LBIST'ed core so that the MISR can generate a proper signature.

Figure 14-1. LBIST conceptual diagram



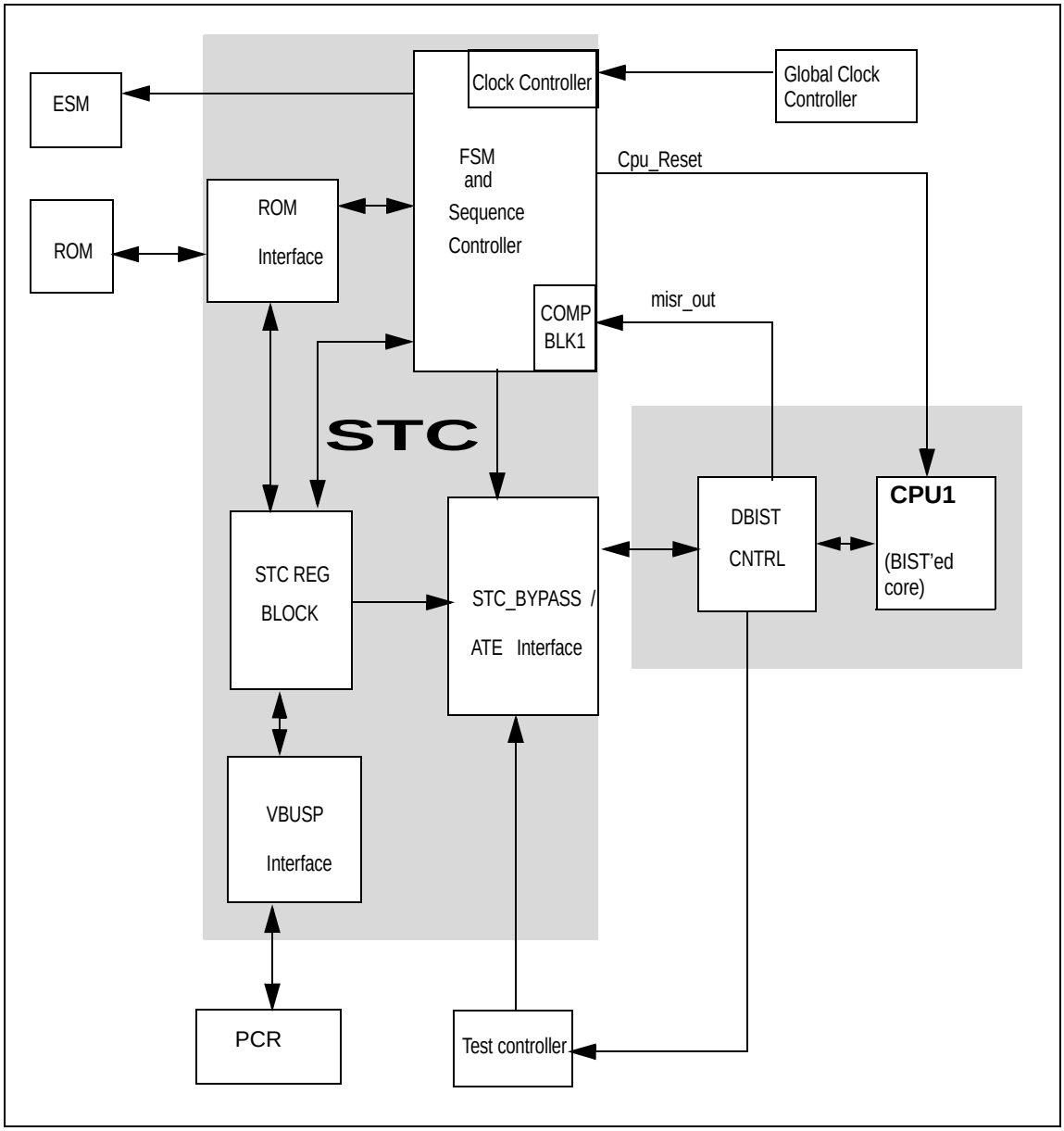
14.3 STC Block diagram

STC module provides an interface to the LBIST controller implemented on the core.

The CPU STC is composed of 5 blocks of logic.

- ROM Interface
- FSM and Sequence control
- Register file
- VBUSP Interface
- STC Bypass/ATE interface

Figure 14-2. Block diagram for Single CPU architecture



14.4 Module Description

14.4.1 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 is stored in ROM.

14.4.2 FSM and Sequence control

This block generates the signals and data to LBIST controller based on the seed, test_type, scan chain depth. The sequence of operation per interval are defined in the Flow chart in Section 14.5. The timing protocol and flow diagrams are covered in Section 14.11.

14.4.2.1 Clock Control

The CLOCK CNTRL sub-block is handles the clock selection and clock generation for ROM, LBIST controller as well as BIST'ed core (CPU) clocks.

14.4.3 Register Block

14.4.3.1 Control Registers

This block handles the control of the Self Test Controller. It includes control registers to specify the test model type like Stuck-at or transition delay methodologies. This also controls the reseeding (Reloading the existing seed of the PRPG) in the LBIST controller.

This also handles the buffering the LBIST seed data and pipelines according to the number of shadow scan chains.

14.4.3.2 Configuration Registers

This contains various configuration and status registers which provide the result of selftest run. These registers are memory mapped and accessible through VBUSP interface.

14.4.4 STC Bypass / ATE Interface

This is a production test interface. This module allows bypassing the self test FSM. The LBIST signal interface are brought out directly to the module ports and these are accessible to ATE (tester) at the device level. The intent on the block is to provide capability for fault isolation for parts failing the CPU self test run.

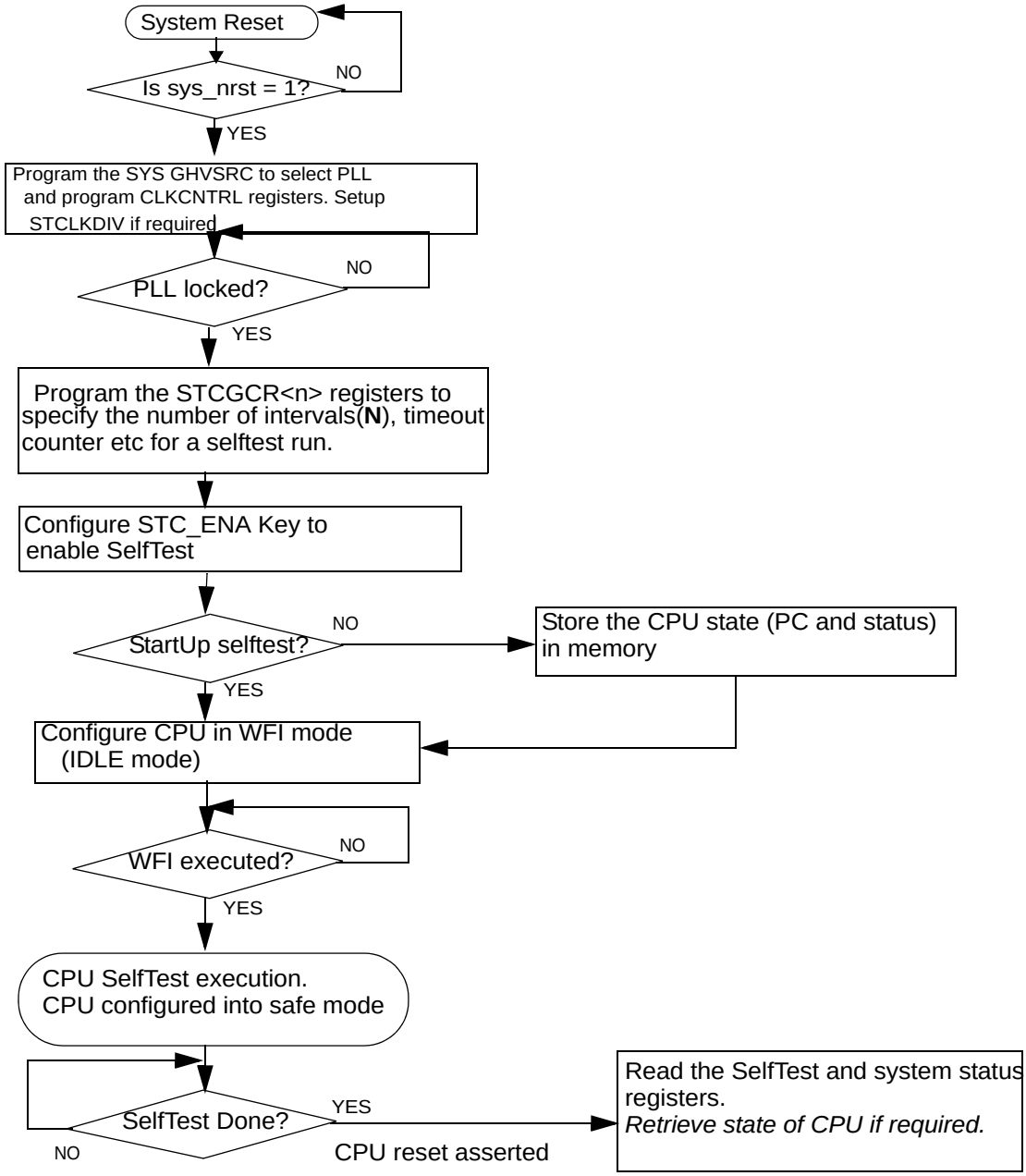
This modules receives two sets of LBIST signals; one from device test controller and another similar set from self test FSM (test sequencer). The selftest enable key is used to select one of the above mentioned datapath to the CPU LBIST controller.

14.4.5 VBUSP Interface

STC control registers are accessed through VBUSP interface. During application programming, configuration registers are programmed through the VBUSP interface, to enable and run the self test controller.

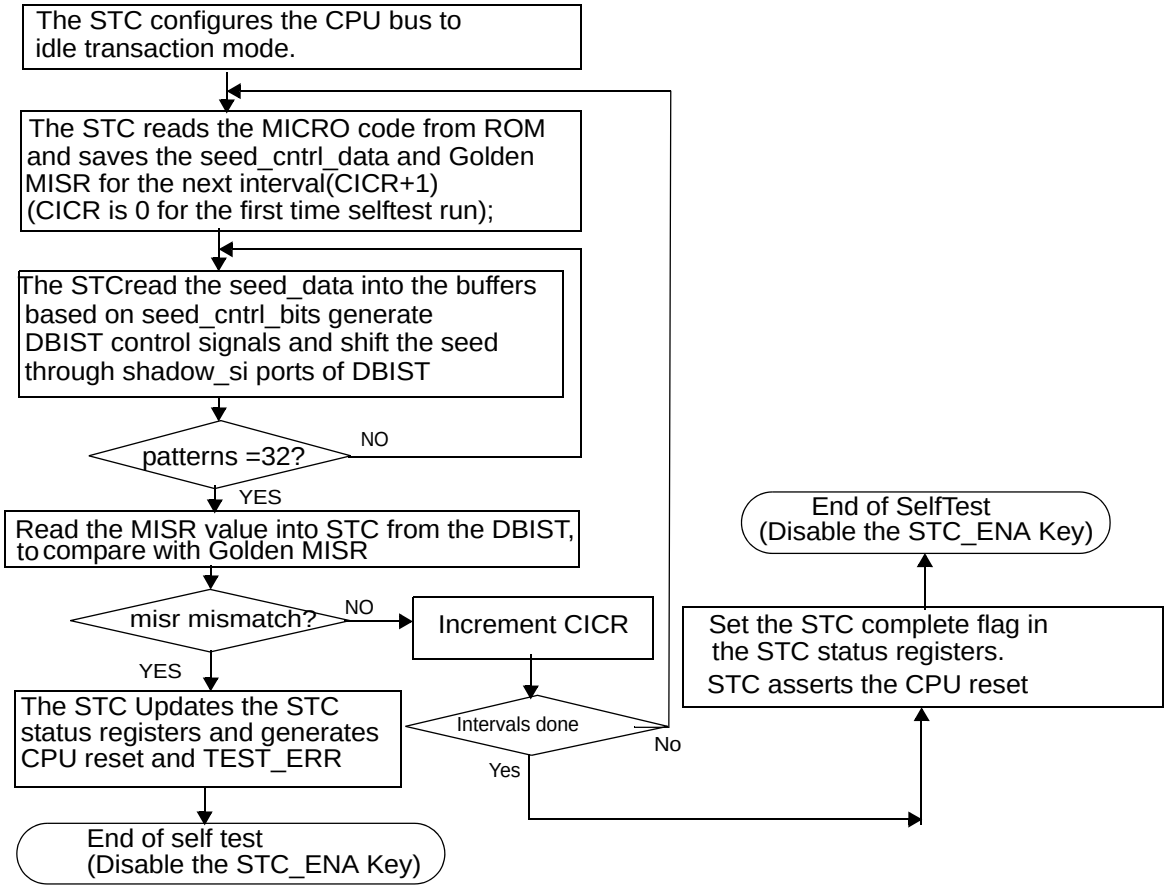
14.5 Application Self Test Flow Chart

Figure 14-3. Application Self Test Flow Chart



14.6 SelfTest Execution Flow

Figure 14-4. SelfTest Execution Flow



14.7 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 DBIST controller is shifted into the STC. This is compared with the golden MISR value stored in the ROM.

At the end of CPU self test, the STC controller updates the status flags in Global Status Register (STCGSTAT) and resets the CPU. In case of a MISR mismatch or a test timeout, error signals are generated to ESM module. TEST_ERR signal is asserted, when a MISR miscompare occurs during the self test. TIMEOUT_ERR is asserted when a time out occurs during the self test, meaning the test could not complete within the time specified in Timeout counter Preload register STCTPR. However at the device level, these two errors may be combined and mapped to a single ESM channel.

Note:
With the completion of the STC and the CPU reset, the nVIC is also reset, not the M3VIM. As a result, both run out of sync. As a consequence, STC can not be run out of a ISR.

14.8 STC clock Divider

System module provides a pre-scalar (STCLKDIV) for STC clock. STC clock is derived from system clock (HCLK) and this can be pre-scaled to achieve a lower STC clock frequency. This implementation provides flexibility to reduce the current consumption during LBIST execution without having to lower the HCLK frequency. An effect of lowering the frequency is that LBIST execution time is increased. The trade-off between current consumption and LBIST execution time needs to be considered.

For asynchronous dual core systems, a separate divider is implemented for each core in the respective system modules.

14.9 Control Registers

STC control registers are accessed through VBUSP interface. Read and write access in 8,16 and 32 bit are supported. The base address for the control registers is 0xFFFF E400.

All reserved bits are read as zeros. Write to reserved fields has no effect.

Table 14-1. 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
0x00 STCGCR0 Page 730	INTCOUNT															
	Reserved															RS_CNT
0x04 STCGCR Page 731	Reserved															
	Reserved															STC_ENA
0x08 STCTPR Page 732	RTOD[31:16]															
	RTOD[15:0]															
0x0C STC_CADDR Page 733	ADDR[31:16]															
	ADDR[15:0]															
0x10 STCCICR Page 734	Reserved															
	N[15:0]															
0x14 STCGSTAT Page 735	Reserved															
	Reserved														TEST_FAIL	TEST_DONE
0x18 STCFSTAT Page 736	Reserved															
	Reserved													TO_ERR	Reserved	CPU1_FAIL

Table 14-1. Registers (Continued)

0x2C CPU1_CURMISR3 Page 737	MISR[31:16]
	MISR[15:0]
0x30 CPU1_CURMISR2 Page 737	MISR[63:48]
	MISR[47:32]
0x34 CPU1_CURMISR1 Page 737	MISR[95:80]
	MISR[79:64]
0x38 CPU1_CURMISR0 Page 737	MISR[127:112]
	MISR[111:96]

14.9.1 Suspend mode consideration:

In suspend mode all register write accesses to the privilege mode only write registers can also be performed in user mode

14.9.2 STC global control register0 (STCGCR0)

This register is described in Figure 14-5 and Table 14-2. The offset address is 0x00.

Figure 14-5. STC global control register0 (STCGCR0)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INTCOUNT															
RWP -1															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															RS_CNT
RWP -1															RWP -0

U = Undefined; R = Read, WP = Write in privilege mode,-*n* = Value after reset

Table 14-2. STC global control register0 (STCGCR0) Field Descriptions

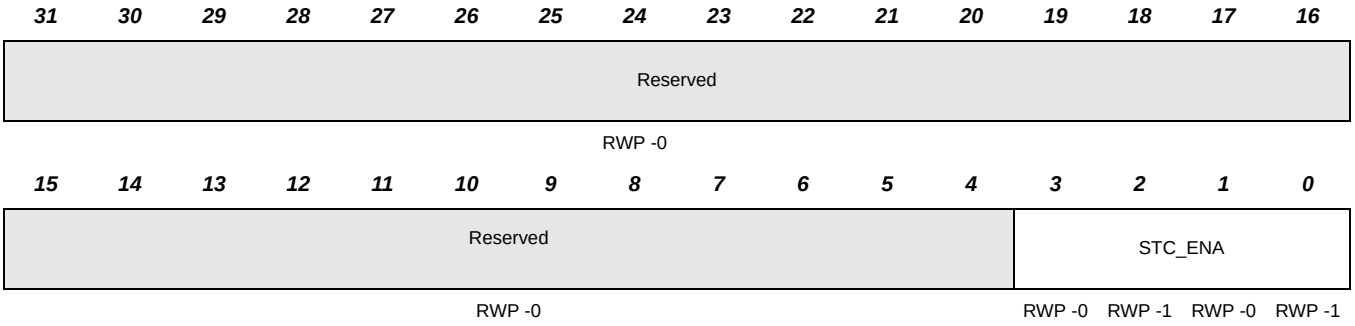
Bit	Name	Value	Description
31–16	INTCOUNT		Number of intervals of selftest run This register specifies the number of intervals to run for the selftest run. This correspond to the number of intervals to be run from the value reflected in the current interval counter.
15-1	Reserved		Reads return 0 and writes have no effect.
0	RS_CNT		Restart or Continue This bit specifies the selftest controller whether to continue the run from next interval onwards or to restart from interval 0. This bit gets reset after the completion of selftest run.
		0	Continue STC run from previous interval.
		1	Restart STC run from interval 0.

Note:
On a powerup reset or system reset this register gets reset to its default values.

14.9.3 STC Global Control Register1 (STCGCR1)

This register is described in Figure 14-6 and Table 14-3. The offset address is 0x04.

Figure 14-6. STCGCR1 (STC Global Control Register1)



U = Undefined; R = Read, WP = Write in privilege mode,-*n* = Value after nPORST (power on reset) or System reset

Table 14-3. STCGCR1 (STC Global Control Register1) Field Descriptions

Bit	Name	Value	Description
31–4	Reserved		Reads return 0 and writes have no effect.
3-1	STC_ENA		Self test run enable key
		1010	Self test run enabled.
		all others	Self test run disabled

Note:
On a powerup 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.

14.9.4 Self Test Run Timeout Counter Preload Register (STCTPR)

This register is described in Figure 14-7 and Table 14-4. The offset address is 0x08.

Figure 14-7. Self Test Run Timeout Counter Preload Register (STCTPR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RTOD[31:16]															
RWP-1															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RTOD[15:0]															
RWP -1															

U = Undefined; R = Read, WP = Write in privilege mode,-*n* = Value after nPORST (power on reset or system reset)

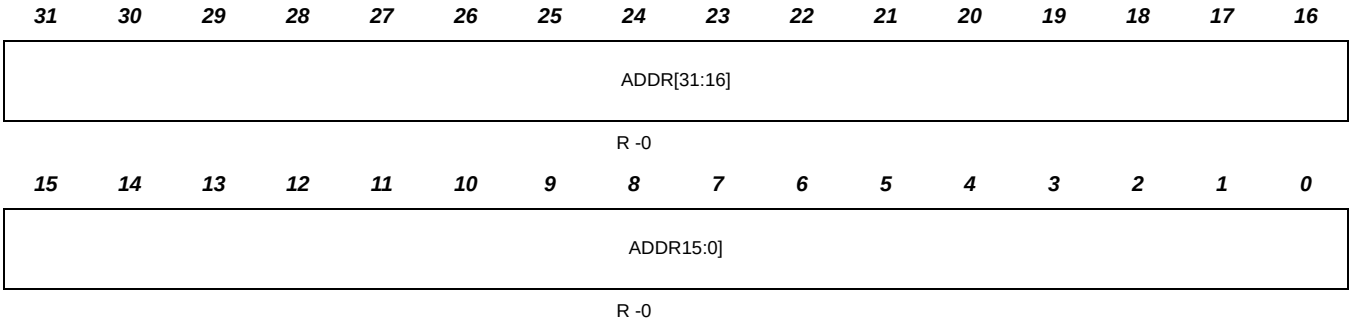
Table 14-4. Self Test Run Timeout Counter Preload Register (STCTPR)

Bit	Name	Value	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 (TIMEOUT_ERR) will be triggered after the initiation of the self test run. This is a fail safe feature to not hang-up the system on account of any run away self test issues, The above 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: This register gets reset to its default value with Power on or system reset assertion.**

14.9.5 STC Current ROM Address Register (STC_CADDR)

This register is described in Figure 14-8 and Table 14-5. The offset address is 0x0C.

Figure 14-8. STC Current ROM Address Register (STC_CADDR)



U = Undefined; R = Read, WP = Write in privilege mode,-*n* = Value after nPORST (power on reset) or system reset

Table 14-5. STC Current ROM Address Register (STC_CADDR) Field Descriptions

Bit	Name	Value	Description
31–0	ADDR		Current ROM Address This register reflects the current address ROM address (for micro code load) which is the current value of the STC program counter.

Note:
When the global configuration register restart bit STCGCR0.0 is set to a 1 on the start of a self test run or on a powerup reset or system reset this register resets to all zeroes.

14.9.6 STC Current Interval Count Register (STCCICR)

This register is described in Figure 14-9 and Table 14-6. The offset address is 0x10.

Figure 14-9. STC Current Interval Count Register (STCCICR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R -0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
N[15:0]															
R -0															

U = Undefined; R = Read, WP = Write in privilege mode,-*n* = Value after reset

Table 14-6. STC Current Interval Count Register (STCCICR) Field Descriptions

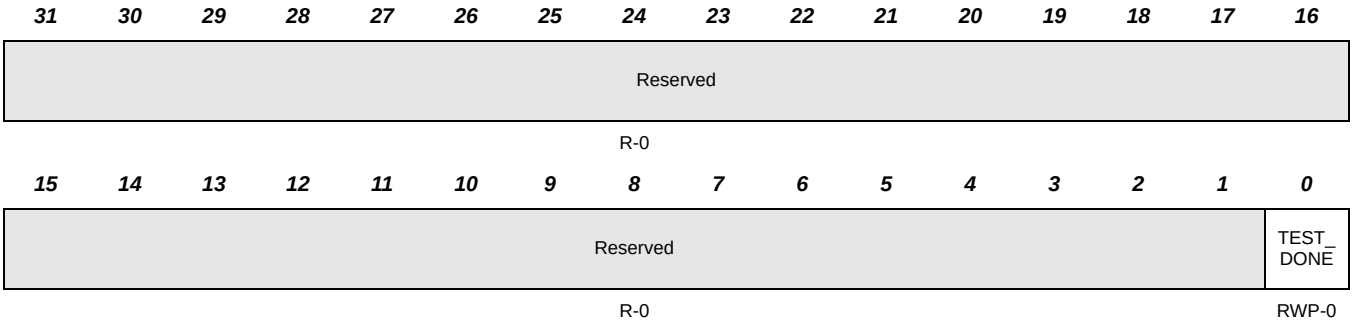
Bit	Name	Value	Description
31–16	Reserved		Reads return 0 and writes have no effect.
15-0	N		Interval Number This specifies the Last executed Interval number.

Note:
When the global configuration register restart bit STCGCR0.0 is set to a 1 or on a powerup reset the Current interval counter resets to default value else it always maintains its value.

14.9.7 SelfTest Global Status Register (STCGSTAT)

This register is described in Figure 14-10 and Table 14-7. The offset address is 0x14.

Figure 14-10. SelfTest Global Status Register (STCGSTAT)



U = Undefined; R = Read, WP = Write in privilege mode,-*n* = Value after reset

Table 14-7. SelfTest Global Status Register (STCGSTAT) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved		Reads return 0 and writes have no effect.
0	TEST_DONE		Test Done
		0	Not completed
		1	Self test run completed
			Note: The above status bit can be cleared to its default value on a write 1 to the above bit. Additionally when the STC_ENA Key is enabled from a non 1010 value to a 1010 value the above status flag gets cleared to its default value. 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. An reset is generated to the CPU on which the STC run is being performed when TEST_DONE goes high (The test is completed). This register gets reset to its default value with Power on reset assertion.

14.9.8 SelfTest Fail Status Register (STCFSTAT)

This register is described in Figure 14-11 and Table 14-8. The offset address is 0x18.

Figure 14-11. SelfTest Fail Status Register (STCFSTAT)

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													TO_ERR	Reserved	CPU1_FAIL
R -0													RWP-0		RWP-0

U = Undefined; R = Read, WP = Write in privilege mode,-*n* = Value after nPORST or system reset

Table 14-8. SelfTest Fail Status Register (STCFSTAT) Field Descriptions

Bit	Name	Value	Description
31–3	Reserved		Reads return 0 and writes have no effect.
2	TO_ERR		Timeout Error
		0	No time out error occurred
		1	SelfTest run failed due to a timeout error
1	Reserved		Reads return 0 and writes have no effect.
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 above 3 status bits can be cleared to their default values on a write 1 to the above bits. Additionally when the STC_ENA Key is enabled from a non 1010 value to a 1010 value the above 3 status flags get cleared to their default values.
This register gets reset to its default value with Power on reset assertion.

14.9.9 CPU1 Current MISR Register (CPU1_CURMISR[3:0])

This register is described in Figure 14-12 through Figure 14-15 and Table 14-9. The offset address for CPU1_CURMISR3 is 0x2C.

Figure 14-12. CPU1 Current MISR Register (CPU1_CURMISR3)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MISR[31:16]															
R -0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MISR[15:0]															
R -0															

U = Undefined; R = Read, WP = Write in privilege mode,-*n* = Value after reset

The offset address for CPU1_CURMISR2 is 0x30.

Figure 14-13. CPU1 Current MISR Register (CPU1_CURMISR2)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MISR[63:48]															
R -0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MISR[47:32]															
R -0															

U = Undefined; R = Read, WP = Write in privilege mode,-*n* = Value after reset

The offset address for CPU1_CURMISR1 is 0x34.

Figure 14-14. CPU1 Current MISR Register (CPU1_CURMISR1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MISR[95:80]															
R -0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MISR[79:64]															
R -0															

U = Undefined; R = Read, WP = Write in privilege mode,-*n* = Value after reset

The offset address for CPU1_CURMISR0 is 0x38.

Figure 14-15. CPU1 Current MISR Register (CPU1_CURMISR0)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MISR[127:112]															
R -0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MISR[111:96]															
R -0															

U = Undefined; R = Read, WP = Write in privilege mode,-*n* = Value after reset

Table 14-9. CPU1 Current MISR Register (CPU1_CURMISR[3:0]) Field Descriptions

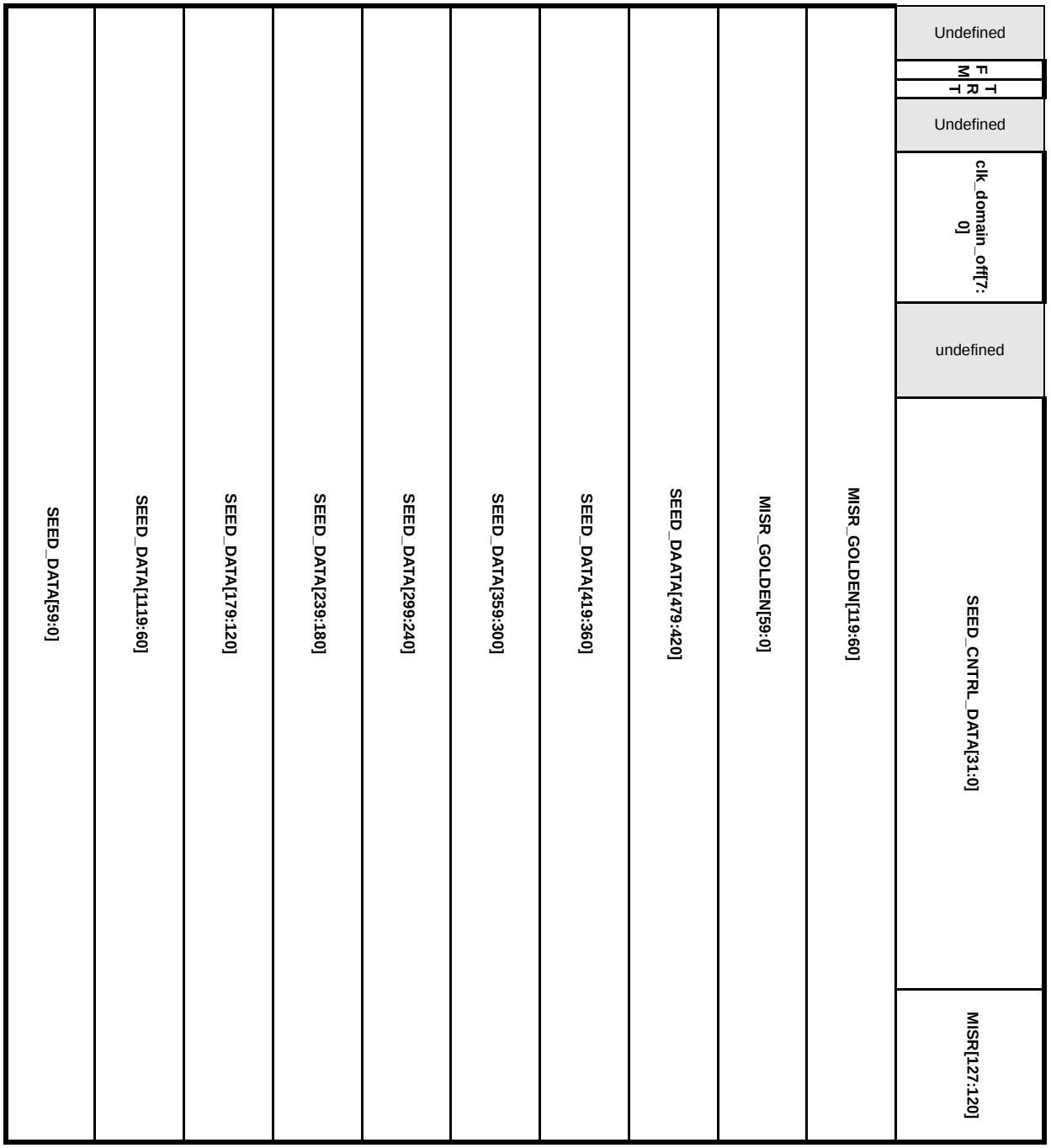
Bit	Name	Value	Description
127-0	MISR		MISR data from CPU1 This register contains the MISR data from the CPU1 for the current interval. This value will be compared with the GOLDEN MISR value copied from ROM.

Note:
This register gets reset to its default value with Power on or system reset assertion.

Note:
This register reads are not reliable. Only 8 bits are loaded into the shift register instead of 128.

14.10 ROM Organization

Figure 14-16. ROM Organization for a Interval (Their might be a maximum of 32 Seed_datas/interval)



The ROM contains the data to be processed by STC for the self test run. This includes the SEED_DATA, Golden MISR compare and the interval specific configuration data for the STC and the DBIST controller. The ROM space is divided into chunks and each chunk contains the data corresponding to one DBIST interval. The size required for each interval may vary depending on the number of seeds required for that particular interval. The first selftest run starts reading the ROM from address 0. Then STC continues to read the ROM

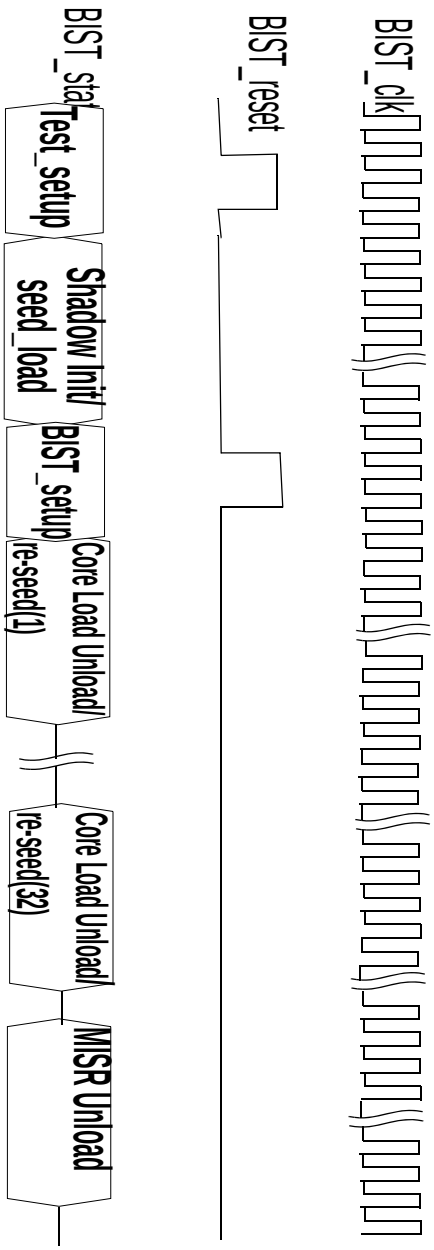
till it finishes the particular selftest of ‘N’ intervals. STC retains its ROM address (STC_CADDR) for the next selftest run unless there is power on reset or an self test restart.

Golden MISR contains the Golden signature data of the current Interval. This value is used to compare with actual MISR value to generate the pass/fail information of the interval.

14.11 Timing Diagrams

14.11.1 BIST Operation: For a self test run for One Interval

Figure 14-17. BIST Operation



The above diagram shows the protocol for a self test run with one interval:

1. Test setup procedure: Once the self test run is initiated and the interval configuration and MISR values are stored in the STC an reset is generated for 2 controller clock cycles to clear the scan_en counter of the BIST controller to its reset state.
2. Shadow Init seed load procedure: The shadow PRPG of the BIST controller is loaded with its initial seed value by the STC from the ROM.
   The seed in the shadow PRPG is loaded into the PRPG at the end of this procedure

3. BIST setup procedure: An reset is generated to the BIST controller to clear the values of the BIST controllers MISR
4. Core Load_unload procedure: While the PRPG is generating the scan patterns to the BISTe'd core the next seed value is loaded into the shadow PRPG (if re-seed = 1) else the shadow PRPG maintains its previous value. Step 4 is repeated for 32 times which constitutes an interval.
5. The BIST controller scan's out the MISR value to the STC on which it is compared with its golden value to indicate the pass or fail of an interval.

Clock considerations:

1. During a pattern shift (At-speed and Stuck-At) the slowest clock of the BISTe'd core is activated as the shift clock for all the clock domains in the BISTe'd core.
2. For an Stuck-At interval the slowest clock of the BISTe'd core is the source for all the functional clock domains of the BISTe'd core during capture.
3. Captures for At-speed interval are done per individual BISTe'd core domain clock during which all other clock domains are gated off.

These GTM interface signals are muxed with the STC FSM generated respective signals in the Bypass Block.

Analog To Digital Converter (ADC) Module

Topic		Page
15.1	Overview	744
15.2	Introduction	745
15.3	Basic Features and Usage of the ADC	747
15.4	Advanced Conversion Group Configuration Options	755
15.5	ADC Module Interrupts	758
15.6	ADC Error Calibration	761
15.7	ADC Built In Diagnostics and Self Test Logic	764
15.8	ADC Special Modes	768
15.9	ADEVT Pin General Purpose I/O Functionality	769
15.10	ADC Control Registers	771

15.1 Overview

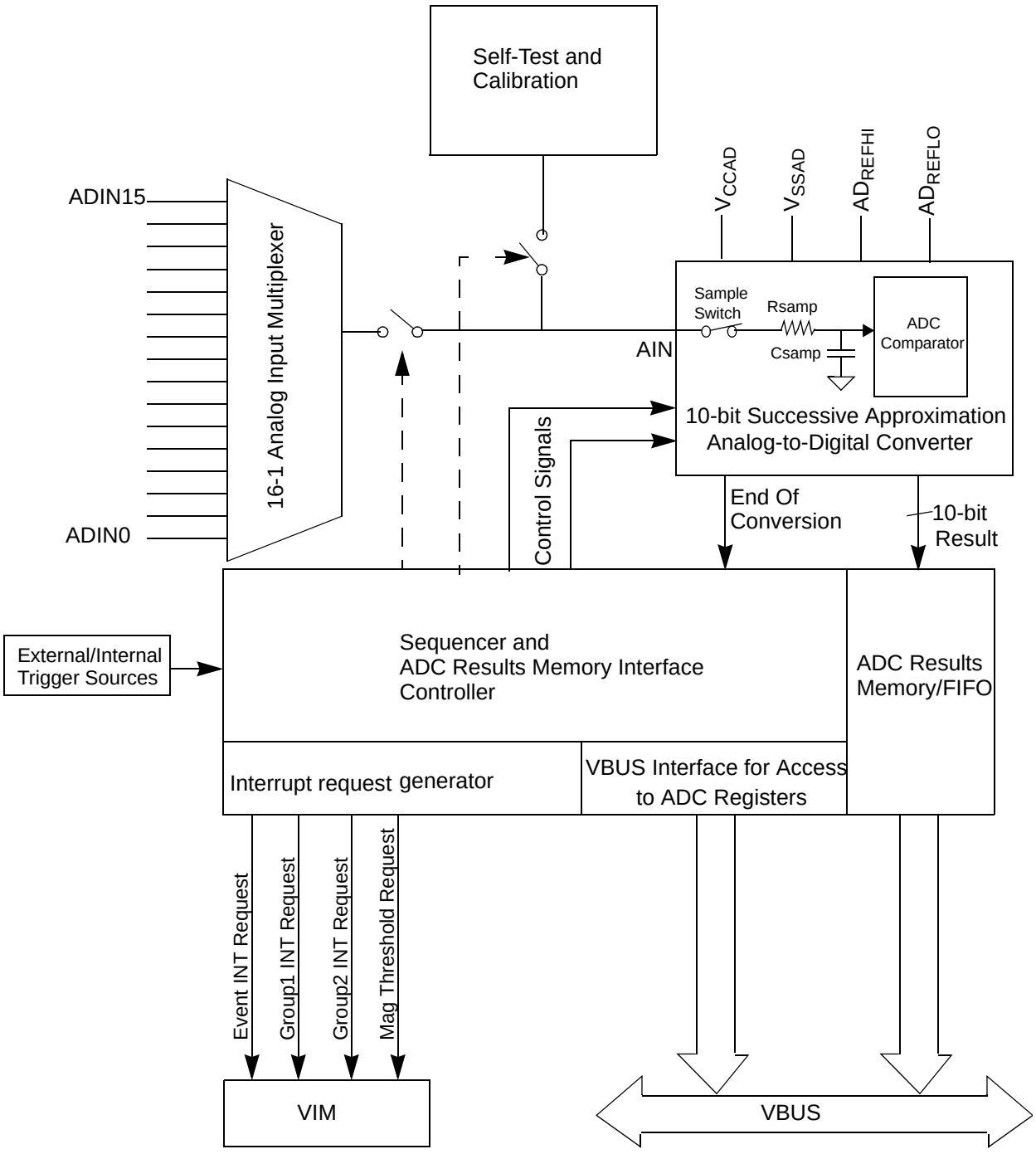
The main features of the 10-bit ADC implemented in this device include:

- 10-bit resolution
- Successive-approximation-register (SAR) architecture
- Three conversion groups: Group1, Group2 and Event Group
- Software and Hardware triggered Event Groups
- 64 word ADC memory/FIFO
  - Size of memory regions for each conversion group is programmable
  - Provides parity protection
- Single and Continuous conversion modes
- Embedded self-test and calibration logic
- External event pin (ADEVT) programmable as general-purpose I/O

15.2 Introduction

This section presents a brief functional description of the analog-to-digital converter (ADC) module. Figure 15-1 illustrates the components of the ADC module.

Figure 15-1. ADC Block Diagram



15.2.1 Analog Input Multiplexer

The analog input multiplexer (MUX) connects the selected input channel to the AIN input of the ADC core. The ADC module supports up to 16 inputs as shown in Figure 15-1 . The sequencer selects the channel to be converted.

15.2.2 Self-Test and Calibration

The ADC includes specific hardware for detecting open/shorts on an ADC analog input pin. It also allows the application program to calibrate the ADC. The self-test mode and the calibration modes are controlled by the sequencer. For more details, see the Section 15.6 and Section 15.7.2.

15.2.3 Analog-to-Digital Converter Core

The analog conversion range is determined by the reference voltages: AD_{REFHI} and AD_{REFLO}. Both AD_{REFHI} and AD_{REFLO} must be chosen not to exceed the analog power supplies: V_{CCAD} and V_{SSAD}. Input voltages between AD_{REFHI} and AD_{REFLO} produce a conversion result given by (EQ 1):

$$DigitalResult = \frac{1024x(InputVoltage - ADREFLO)}{(ADREFHI - ADREFLO)} \quad \text{(EQ 1)}$$

15.2.4 Sequencer

The sequencer coordinates the operations of the ADC, including the analog input multiplexer, 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.

The logic handles CPU read/write operations, interrupts, halts, resets, memory allocation and handling. The logic in this block also supports self-test mode and calibration operations. This logic block generates the internal ADC clock that controls the ADC timing.

All the features of the sequencer are discussed in detail in the following sections of this document.

15.2.5 Conversion groups

The ADC module can service one of three conversion groups at any time: Group1, Group2, and the Event Group. The Group1 and Group2 are software-triggered by default and can be configured as hardware triggered (ADG1MODECR.3, ADG2MODECR.3) while the Event Group is hardware triggered only. Each conversion group has a separated set of registers to:

- Select the input channel,
- Configure in single or continuous conversion mode,
- Set acquisition time,
- Read conversion data,
- Handle the group conversion end interrupt, group memory threshold interrupt and group memory overrun interrupt.

15.3 Basic Features and Usage of the ADC

Figure 15-2 - Figure 15-4 shows how to start AD conversions after reset. Figure 15-2 describes how to initialize the ADC module; Figure 15-3 describes how to start/trigger single conversion; And Figure 15-4 describes how to start/trigger continuous conversion. The function in these flow charts will be described in details in following sub-sections.

Figure 15-2. ADC Initialization Flow Chart

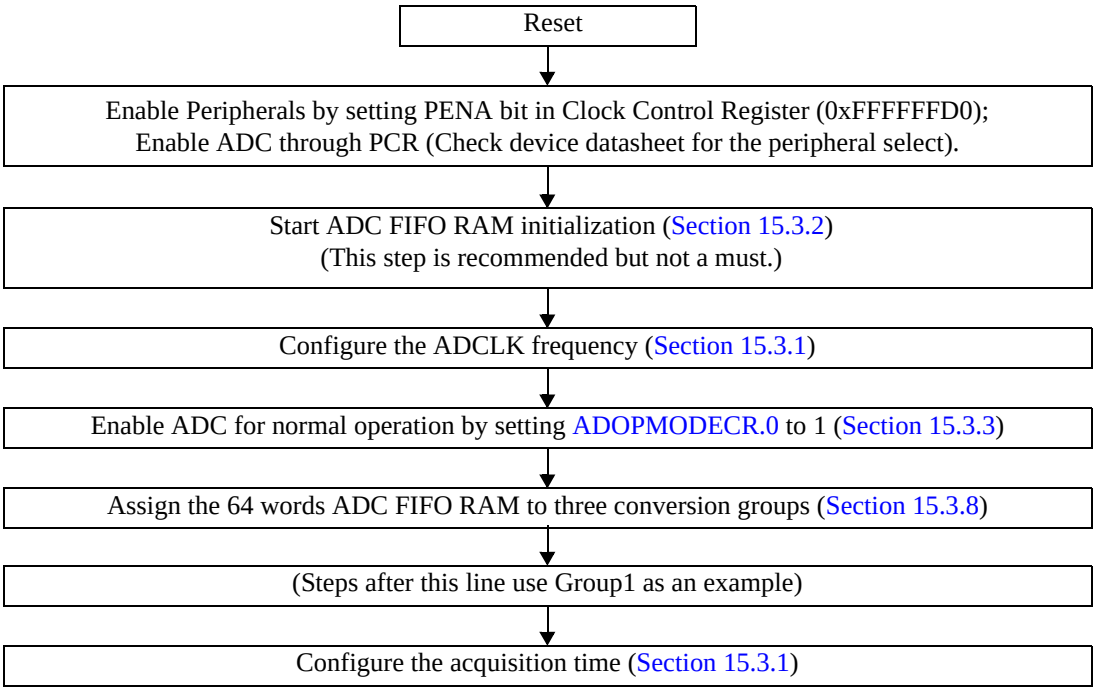
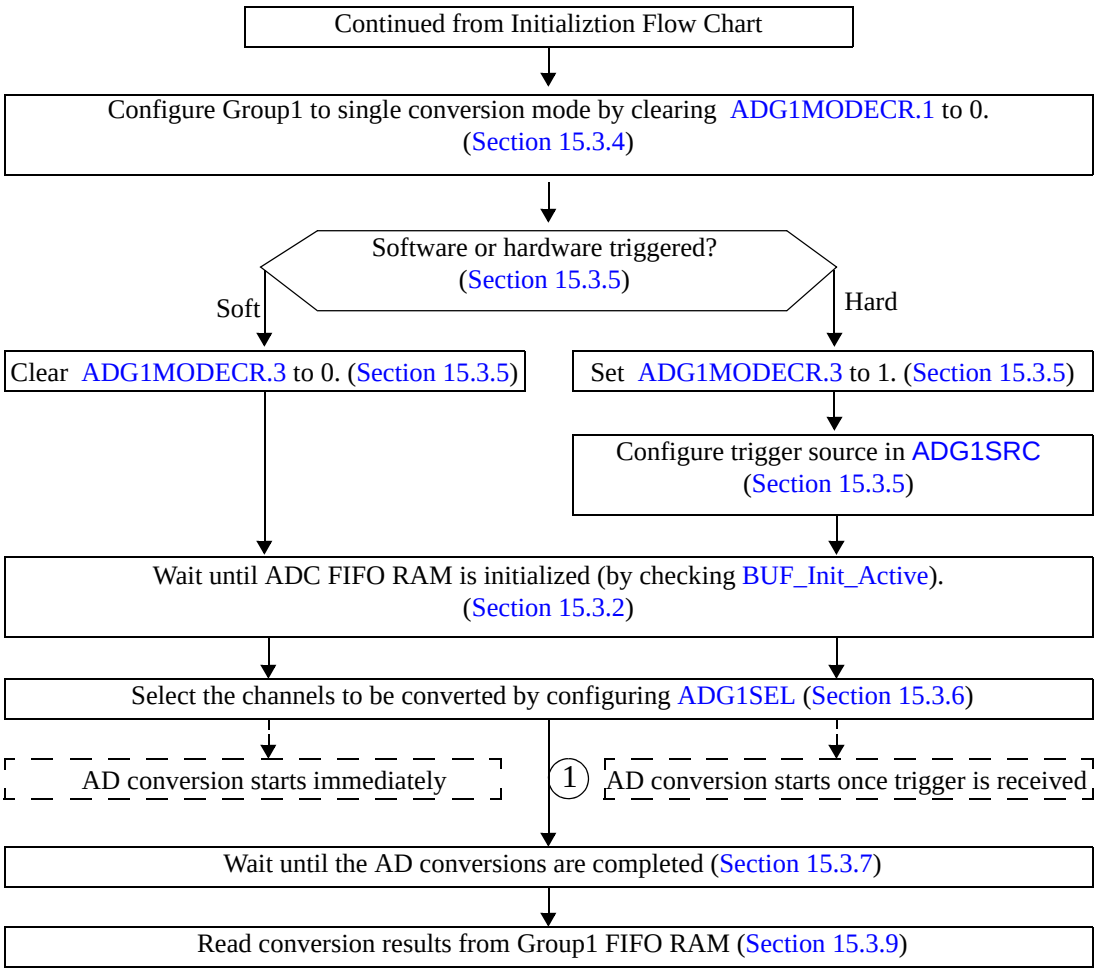
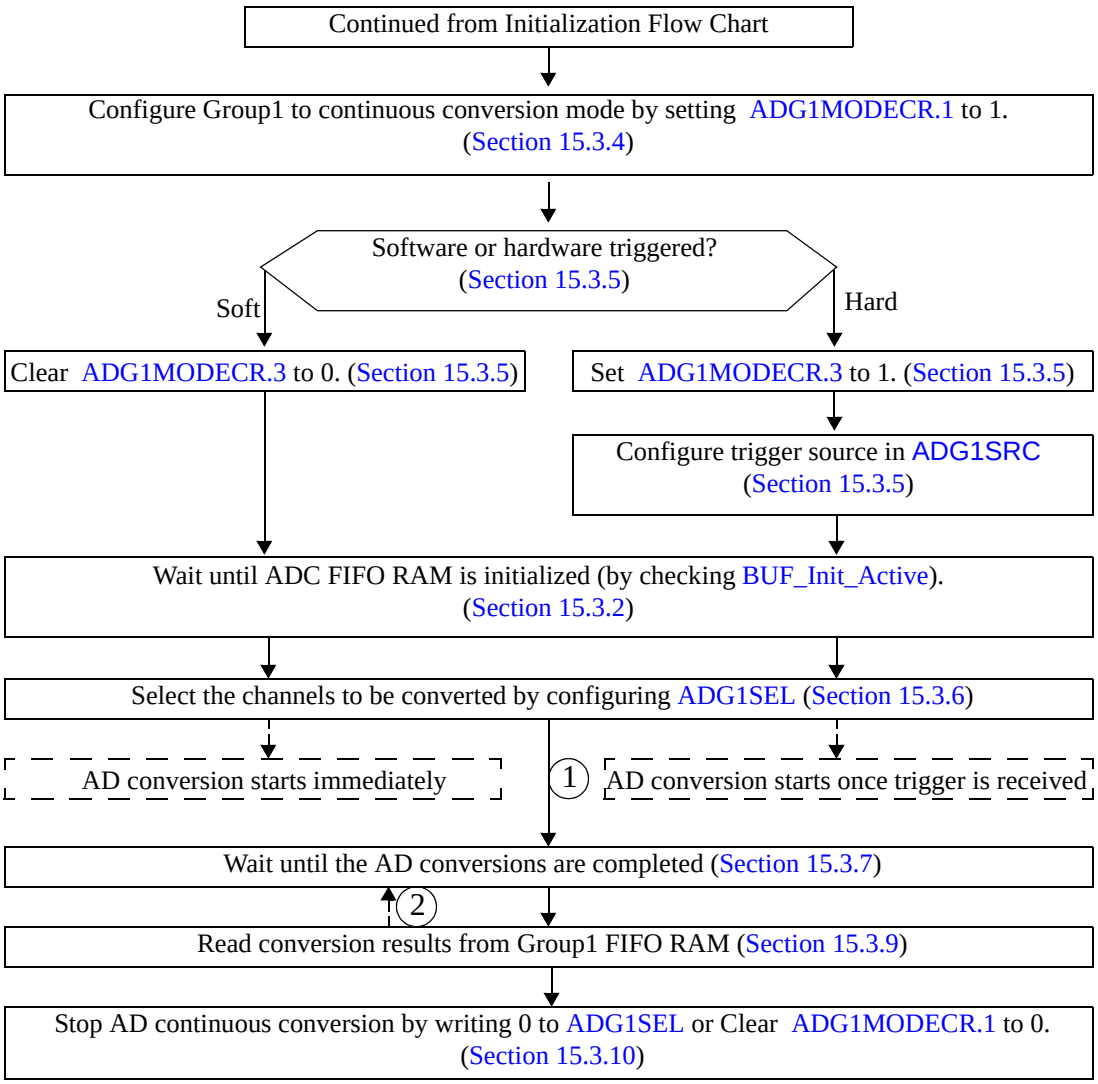


Figure 15-3. Flow Chart for Single Conversion



1. If hardware triggered, ADC core must wait for the trigger signal before conversion starts.

Figure 15-4. Flow Chart for Continuous Conversion



1. If hardware triggered, ADC core must wait for the trigger signal before conversion starts.
2. In continuous conversion mode, ADC will keep AD converting, writing FIFO and set ADG1SR.0.

15.3.1 How to setup the ADCLK speed and the acquisition time?

The clock for the ADC core is ADCLK. 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 in the ADCLOCKCR, bits [4:0], is used to divide down the VCLK by 1 up to 32. The ADCLK frequency range is specified in device datasheet.

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 4097 ADCLK cycles.

For example, Group1 acquisition time, t_{ACQG1} = ADG1SAMP[11:0] + 2, in ADCLK cycles.

15.3.2 How to initialize ADC Results FIFO RAM?

The ADC module allows the application to auto-initialize the ADC results FIFO RAM to all zeros and fill in the parity bits. To avoid parity errors when reading ADC RAM location without AD Conversion data, it is recommended to initialize the ADC RAM. 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 1010b to the bits [3:0] of the MINITGCR register of the System module.
2. Set the control bit for the ADC results FIFO RAM in the MSIENA 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 ADC results FIFO 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.
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 User Guide for more details on the memory auto-initialization process.

15.3.3 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).

15.3.4 How to configure single or continuous modes?

ADG1MODECR.1, ADG2MODECR.1, and ADEVMODECR.1 are used to select between either single or continuous conversion mode for each of the three groups.

15.3.5 How to configure software or hardware trigger?

There are two trigger modes, software-triggered and hardware event-triggered. If a conversion group is configured to be software-triggered, writing the desired channels to the respective Channel-Select Registers will start the AD conversion. If a conversion group is configured to be hardware triggered, writing the Channel-Select Registers only select the desired channedls to be converted. To start the conversion, the application must provide a trigger event, for example, a rising edge of the ADEVT pin.

The Event Group is hardware triggered only. The Group1 and Group2 operating mode control registers (ADG1MODECR.1 and ADG2MODECR.1) have an extra control bit: HW_TRIG. Setting this bit to one configures the group to be hardware event-triggered while clearing this bit to zero configures the group to be software-triggered, which is the default.

When a group is configured to be hardware triggered, the trigger source is defined for each group in the ADEVSRC, ADG1SRC and the ADG2SRC registers.

15.3.6 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 hardware-triggered only. There are up to eight hardware event trigger sources defined for the ADC module. 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 through ADEVSRC register, 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.

15.3.7 How to know the group conversion is completed?

Each conversion group has a 'conversions ended' bit (ADEVSR.0, ADG1SR.0, ADG2SR.0). This bit will be set when all the channels selected in the group has completed. In continuous conversion mode, since the selected channels are served again and again, this bit will be set again and again.

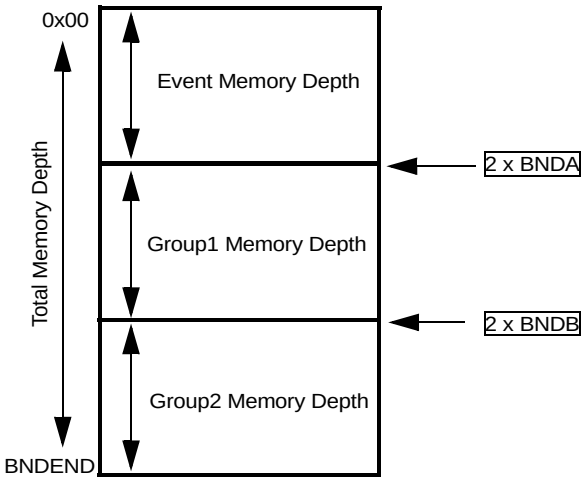
15.3.8 How are results stored in the results memory?

The ADC stores the conversion results in three separate FIFOs in the ADC results FIFO RAM, one FIFO for each group. ADBNDCR contains two 9-bit pointers BNDA and BNDB. They are used to partition the total memory available into three memory regions as shown in Figure 15-5. ADBNDEND contains a 3-bit field called BNDEND that configures the total memory available.

- total FIFO size is 64 32-bit words.

  The user must program BNDEND as 010b (64 words) or less. A reason you would do 'less' is for compatibility with another chip that has less ADC RAM.
- three FIFOs are allocated from the 64 words
- size of EVENT group FIFO is 2 x BNDA words
- size of GROUP1 FIFO is 2 x (BNDB - BNDA) words
- size of GROUP2 FIFO is 2^{BNDEND} x 16 - 2 x BNDB words

Figure 15-5. FIFO Memory Partitioning between Conversion Groups



15.3.9 How to read the results from the results memory?

The CPU can read the conversion results in one of two ways:

a) By reading the group's conversion results from a FIFO queue (ADEVBUFFER, ADG1BUFFER, ADG2BUFFER), or

b) By reading the group's conversion results directly by accessing the correct ADC RAM location.

15.3.9.1 Reading conversion results from the FIFO

The conversion results for each group can be accessed via a range of addresses provided to facilitate the use of the ARM Load-Multiple (LDM) instruction. A single read performed using the LDR instruction can 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 between offset 0x90 and 0xAF (ADEVBUFFER) pulls out one conversion result from the Event Group memory.

Figure 15-6. Format of conversion result read from FIFO, 10-bit ADC

0x90 to 0xAF[1] ADEVBUFFER Page 825	Reserved		
	EV_EMPTY	EV_CHID	EV_DR
0xB0 to 0xCF[2] ADG1BUFFER Page 826	Reserved		
	G1_EMPTY	G1_CHID	G1_DR
0xD0 to 0xEF[3] ADG2BUFFER Page 827	Reserved		
	G2_EMPTY	G2_CHID	G2_DR

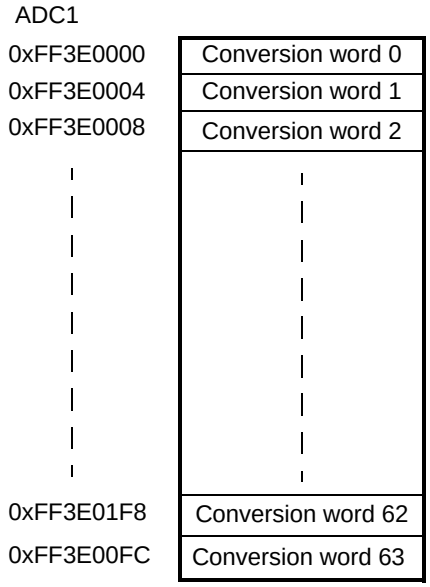
(1) Reading any address within this range accesses the EVENT Group FIFO.
(2) Reading any address within this range accesses the Group1 FIFO.
(3) Reading any address within this range accesses the Group2 FIFO.

Also, for debug purposes, each buffer has an emulation address (ADEVEMUBUFFER, ADG1EMUBUFFER, ADG2EMUBUFFER) that returns the next conversion result from the buffer without removing the result from the buffer itself. For example, reading from offset 0xF0 (ADEVEMUBUFFER) 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.

15.3.9.2 Reading conversion results from the RAM

Figure 15-7 shows the direct mapped view of the ADC result memory. This view may be used instead of the FIFO view if a direct mapped view suits the application better. The table base address is 0xFF3E0000 for ADC. The table has 64 32-bit words result memory.

Figure 15-7. ADC Memory Mapping



The format of the each conversion word is shown in Figure 15-8. ‘Empty’ flag does not exists in the RAM but it does exist during FIFO read.

Figure 15-8. Format of ADC RAM, 10-bit ADC

ADC RAM address	Reserved		
	Res	channel id [3:0]	10-bit conversion result

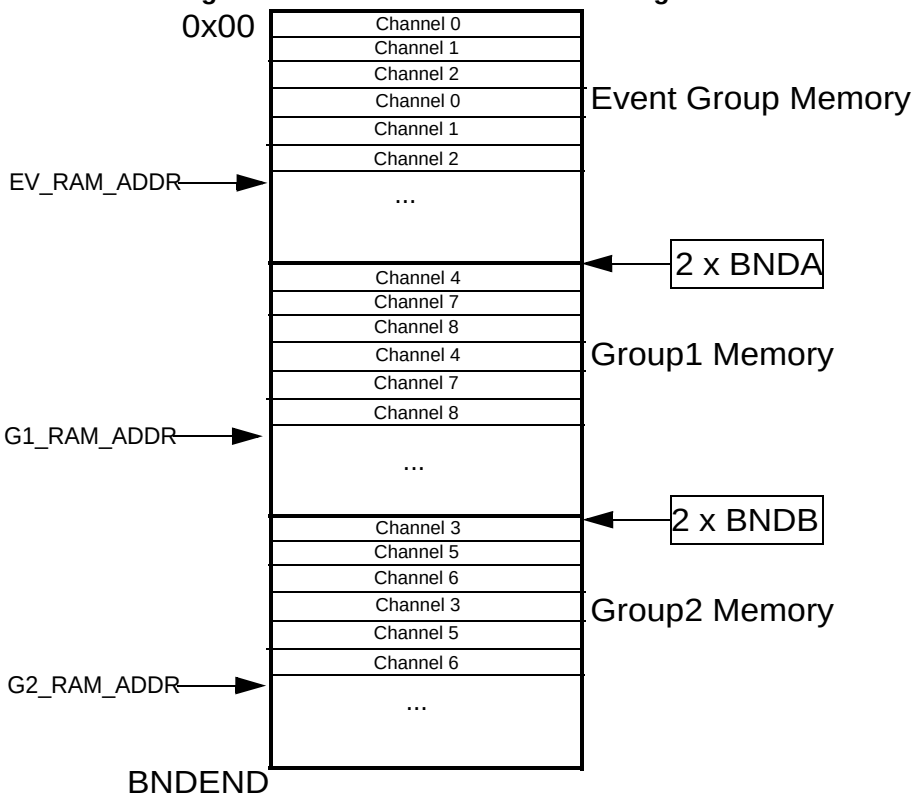
To read conversion results from the RAM directly, the application needs to identify the address ranges for each of the three memory regions for the three conversion groups after performing the segmentation as described in Section 15.3.8. It is up to the application to read the desired results from the three conversion groups. A separate register for each group (ADEVRAMWRADDR, ADG1RAMWRADDR and ADG2RAMWRADDR) holds the ADC buffer index for that group where the ADC will write the next conversion result. For example, if ADG1RAMWRADDR is 0x21, the ADC will write the next conversion result to buffer 33, with the address of (RAM Base Address + 0x21 x 4). The application can therefore calculate how many valid conversion results are available to be read. This mode also allows the application to selectively read the conversion results for any particular input channel of interest without having to read other channels’ conversion results first. The format of the result read directly from RAM is slightly different (no EMPTY bit) from that read from the FIFO interface as shown below.

Please refer to Section 15.3.2 for information about initializing the ADC memory. Please refer to Section 15.7.1 for information about parity check of the ADC memory.

15.3.9.3 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 Group1, and channels 3, 5, and 6 are selected for conversion in Group2. The conversion results will get stored in the three memory regions as shown below:

Figure 15-9. Conversion results storage



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 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 Group1 from the RAM directly. The conversion results for the Group1 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.

15.3.10 How to stop a conversion?

A group’s conversion can be stopped by clearing the group’s channel select register. Writing a non-zero value to the group’s channel select register resets the group’s results FIFO pointer.

In continuous conversion mode, clearing EV_MODE, G1_MODE, and G2_MODE bits will force the corresponding conversion group into single conversion mode. The conversion will stop when all the channels selected in the group has completed.

15.3.11 List of ‘Do NOT’

Do not write to the control registers especially the input-select registers (ADEVSEL, ADG1SEL, ADG2SEL) if there are ongoing AD conversions and the conversion result is needed by the application.

Do not trigger off the ADC RAM auto-initialization process if the ADC module is any of the conversion modes.

Do not write a new conversion result to the results RAM if the ADC Results RAM Test Mode is enabled.

Do not modify the SELF_TEST bit (ADCALCR.24) to enable or disable the self-test mode if there are ongoing AD conversions.

Do not modify the HILO bit (ADCALCR.8) to change the test voltage source if there are ongoing AD conversions.

15.4 Advanced Conversion Group Configuration Options

15.4.1 Single or Continuous Conversion Modes

The EV_MODE, G1_MODE, and G2_MODE bits in the group's operating mode control registers (ADEVMODECR, ADG1MODECR, ADG2MODECR) are used to select between either single or continuous conversion mode for each of the three groups.

15.4.1.1 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, Group1 and 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 (ADEVSR, ADG1SR, ADG2SR). 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.

15.4.1.2 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.

15.4.2 Conversion Group Freeze Capability

Each conversion group can be configured to allow its conversions to be frozen whenever there is a request for conversion in another 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 new request will start after the conversion of the current channel of the active group is complete. 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.

15.4.3 Conversion Group Priority

There's an inherent priority between the conversions groups (EV, G1, G2) - Event Group being the highest priority and Group2 being the lowest priority group. Similarly the channel0 has the highest priority and channel 31 has the lowest priority within a group and it's limited to the group. So, if two groups are triggered at the same time, then the higher priority group will start first, complete and then the next priority group will be addressed.

By default (with FREEZE feature disabled), while a group conversion is underway, no other groups can stop/break the conversions until all the channels in that group are completed even if there are higher priority groups pending.

With "FREEZE" feature enabled for a conversion group, it can be interrupted by a new request on any other conversion groups. By using the FREEZE bit appropriately between the 3 groups, it's possible to maintain priority order of group conversions.

15.4.4 8-bit or 10-bit Result Mode

Some applications can use only the ADC conversion results as 8-bit. The ADC module can automatically shift a group's conversion results right as the result is being read from the FIFO (Section 15.3.9.1). The conversion results read directly from memory is not impacted. This eliminates the need for the application program to shift each conversion result itself, and results in more efficient code execution.

The 8-bit result mode is enabled by setting the G1_8Bit, G2_8Bit, and EV_8Bit in the group's operating mode control registers (ADEVMODECR, ADG1MODECR, ADG2MODECR) control bits to "1".

15.4.5 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 **all** the group's results memory upon an overrun condition. Only then can the ADC continue to write new results to the memory. Otherwise, the latest conversion data will be discarded.

15.4.6 Group Channel Id Storage Option

The channel number is always stored in the FIFO RAM along with the conversion result. But the channel number field can be masked out during reads from the FIFO if CHID bit in the group's operating mode control register (ADEVMODECR, ADG1MODECR, ADG2MODECR) is set.

If the CHID bit is not set, the bits [14:10] are forced to 00000 when the conversion results are read out in FIFO mode (ADEVBUFFER, ADG1BUFFER, ADG2BUFFER) from the group's results memory.

If the CHID bit is set, the bits [14:10] in the group's results memory contain the input channel number will be read out in FIFO mode.

15.4.7 Group Trigger Options

The Group1 and Group2 operating mode control registers (ADG1MODECR, ADG2MODECR) 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.

15.5 ADC Module Interrupts

This section describes the interrupts generated by the ADC module. Each conversion group is capable of three interrupts: Group Conversion End Interrupt, Group Memory Threshold Interrupt and Group Memory Overrun Interrupt. The ADC module also has the capability to generate ADC Magnitude Threshold Interrupts.

15.5.1 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 in the corresponding Group Interrupt Enable Control Register(ADEVINTENA, ADG1INTENA, ADG2INTENA).

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. In the continuous - conversion mode, this interrupt will be triggered again and again since the group's END bit will be set repeatedly.

15.5.2 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 (ADMAGINTxCR) 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 threshold counter is a 16 bit signed integer, with legal values between -256 and +255. Programmer should initialize the threshold counter to the number of results that the FIFO should hold before interrupting the CPU. The threshold register value tracks the level of data within the FIFO. This counter decrements whenever a new result is stored in the FIFO, and increments when a value is read from the FIFO. 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.

15.5.3 Group Memory Overrun Interrupt

Whenever the ADC produces a new result but there is no free storage in the corresponding group memory, an overrun occurs. An overrun interrupt is available for each group. In case of an overrun, the application should follow Section 15.4.5 to configure the ADC.

15.5.4 ADC Magnitude Threshold Interrupts

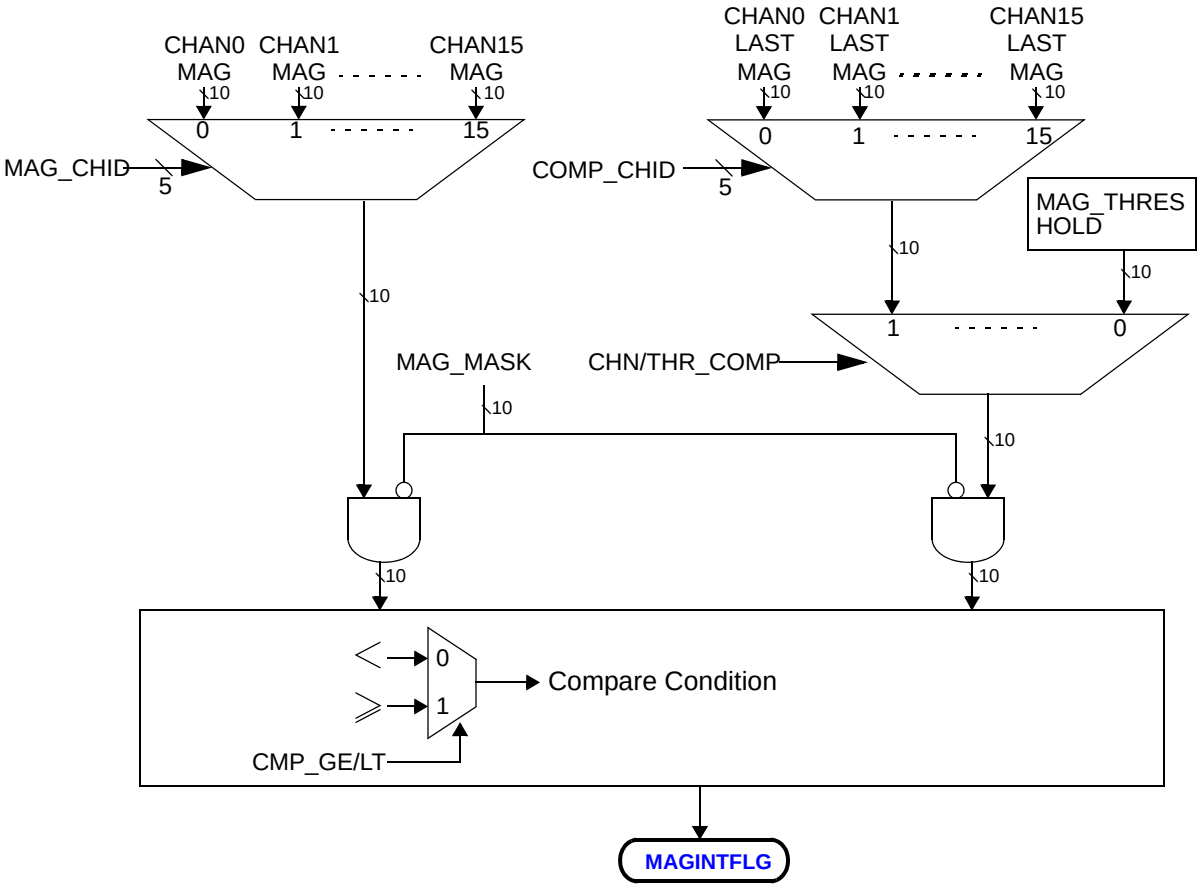
The ADC allows a magnitude threshold interrupt to be generated if the conversion result from a certain channel reaches a predetermined threshold. The predetermined threshold can be a fixed digital value or the last conversion result of channel COMP_CHIDx[4:0]. The interrupt condition can be "greater than or equal to" or "less than". The magnitude comparison can be performed on up to three channels. The comparison parameters are programmed via the Magnitude Threshold Control Register (ADMAGINTxCR). If the conversion result for the selected channel meets the condition programmed, an interrupt is generated.

15.5.4.1 Magnitude Threshold Interrupt Configuration

Figure 15-10 shows how the magnitude threshold interrupts are configured. The following control 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.
2. MAG_CHID: Specifies the channel number from 0 to 15 whose conversion result needs to be monitored.
3. COMP_CHID: Specifies the channel number from 0 to 15 whose last conversion result is used for the comparison with the conversion result of the channel being monitored.
4. MAG_THRESHOLD: 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_THRESHOLD field.
6. MAG_MASK: Specifies the mask for the comparison. The ADC module will mask the corresponding bit in the register ADMAGxMASK for the comparison.
7. MAG_INT_FLG: Specifies whether the magnitude comparison condition is true or false.

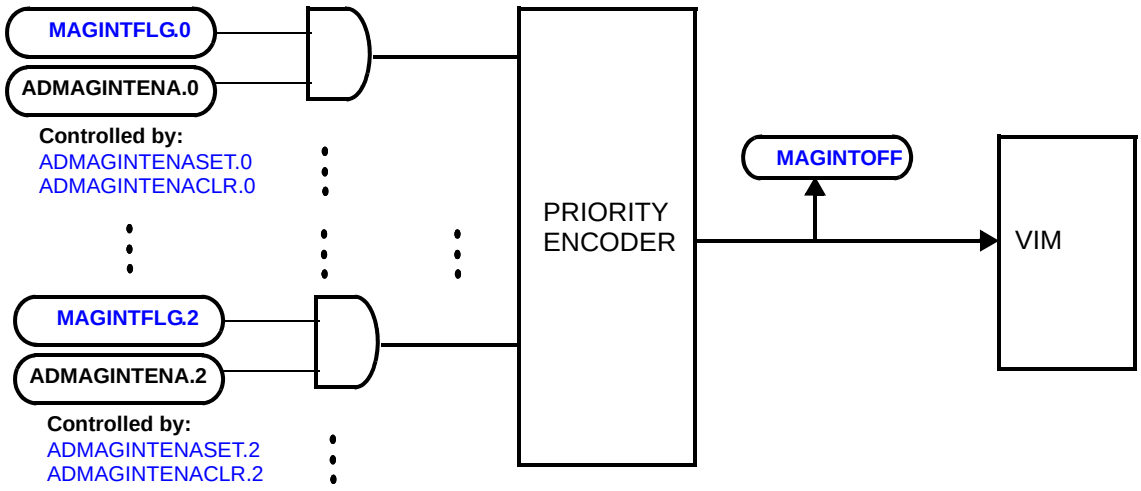
Figure 15-10. Magnitude Threshold Interrupt Configuration



15.5.4.2 Magnitude Threshold Interrupt Generation

Figure 15-11 shows how the magnitude threshold interrupt is generated and sent to VIM (Vector Interrupt Module).

Figure 15-11. Magnitude Threshold Interrupt Generation



Each of the three magnitude interrupts also have separate interrupt enable set and clear 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.

It is possible to have multiple magnitude threshold interrupts pending at the same time. The magnitude threshold interrupt offset register 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.

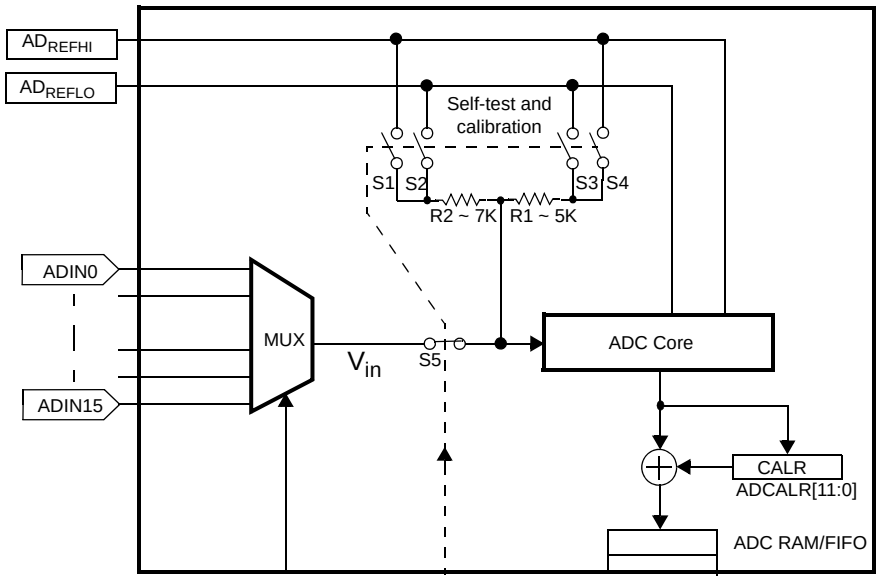
A read from this register updates the register to the next highest-priority pending magnitude threshold interrupt. This read also clears the corresponding flag from the magnitude threshold interrupt flag register.

15.6 ADC Error Calibration

The ADC Error Calibration is used to calculate the offset error and provide a compensation value for normal ADC conversions. 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 RAM.

Figure 15-12 shows the self-test and calibration logic embedded in this device. In normal conversion mode, S5 is closed while S1-S4 are opened. In calibration mode, S5 is opened and the offset error is calculated based on the conversion of the embedded calibration reference voltage.

Figure 15-12. Self-Test and Calibration Logic



15.6.1 Calibration and Offset Error Correction Procedure

The basic calibration routine is as follows:

1. Enable calibration via CAL_EN (ADCALCR.0).

   User must make sure the self-test mode (SELF_TEST, ADCALCR.24) is disabled and no conversion group is being serviced when the calibration mode is enabled.

2. Select the voltage source via BRIDGE_EN and HILO (ADCALCR[9:8]).

   These two bits control the voltage to the calibration reference device shown in Figure 15-12. The positions of the switches in calibration mode are listed in Table 15-1.

Table 15-1. Calibration Reference Voltages†

CAL_EN	BRIDGE_EN	HILO	S1	S2	S3	S4	S5	Reference Voltage
1	0	0	1	0	1	0	0	$(AD_{REFHI}*R1 + AD_{REFLO}*R2) / (R1 + R2)$
1	0	1	0	1	0	1	0	$(AD_{REFLO}*R1 + AD_{REFHI}*R2) / (R1 + R2)$
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}

† The state of the switches in this table assumes that self-test mode is not enabled.

3. Start the conversion by writing a '1' to CAL_ST.

   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).

   Before starting the conversion, the user must make sure the calibration conversion meets the minimum sampling time specification for the ADC. Please see Section 15.3.1 for details.

4. Wait for CAL_ST to go to 0.
5. Get the results from ADCALR and save to memory.

   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.

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 in the 2's complement form with the computed error correction value.

   For example, assume the [D(cal1)+D(cal2)] /2=513 during the Mid-Point Calibration (Section 15.6.2), the ADCALR register should be set to 0x3FF to compensate the offset error.

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.

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.

15.6.2 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 & 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%).

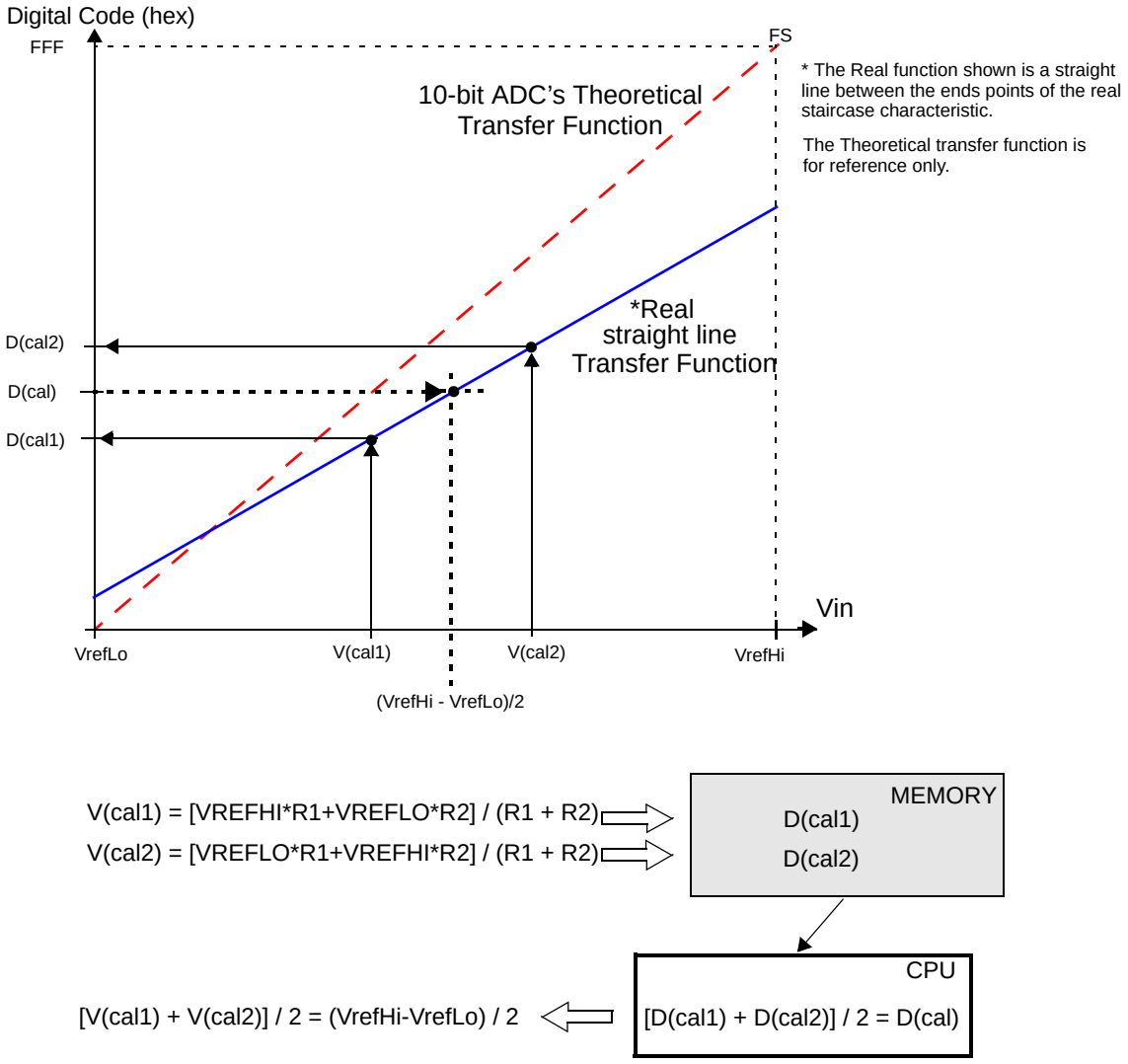
The switched reference voltage device has been specially designed to support a differential measurement of its mid-point voltage. This ensures the accuracy of the mid-point reference, and hence the efficiency of the calibration.

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 15-13 summarizes the mid-point calibration flow.

Figure 15-13. Mid-point value calculation



15.7 ADC Built In Diagnostics and Self Test Logic

15.7.1 ADC RAM Parity and Test

For safety reasons, the ADC RAM has protection by parity to prevent corruption by soft error. The parity scheme is implemented as a continuous background check based on memory access. Section 15.7.1.1 and Section 15.7.1.2 describe how parity works and how to check the scheme of parity.

15.7.1.1 ADC results FIFO RAM Parity

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 ADC RAM parity is controlled by the DEVCR1 register in the system module (address 0xFFFFFFDC). The parity enable is controlled by the ADPARCR register. After reset, the parity is disabled. The application should enable parity check if parity protection is needed.

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 and signals it to the ESM module. 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.

15.7.1.2 Testing the Parity Checking Mechanism

To test the parity checking mechanism itself, the parity RAM has to be made accessible in order to allow manually inserting parity errors. In the defined conversion modes of the ADC, only the ADC module is allowed to write to the results FIFO RAM. A special test mode, ADC results FIFO RAM Test Mode, is defined to allow the application to also write into the ADC results FIFO RAM. Only 32-bit reads and writes are allowed to the ADC results FIFO RAM in this test mode.

Note: Contention on access to ADC results FIFO RAM
The application must ensure that the ADC is not likely to write a new conversion result to the results FIFO RAM when the ADC results FIFO RAM Test Mode is enabled.

The ADC results FIFO RAM Test Mode is enabled by setting the RAM_TEST_EN bit in the ADOPMODECR register.

To mannually insert an error into the parity bits themselves, the parity bits have to be mapped to an different address. Once the TEST bit in the ADPARCR register 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 as shown in Figure 15-14.

Note:
Every conversion word has one parity bit. This parity bit is calculated based on bit 15:0. Bit 31:16 are reserved.

Figure 15-14. Parity Bit Map

ADC1	31 16	0	
0xFF3E0000	See Note	word 0	ADC Conversion Data
0xFF3E0004	See Note	word 1	
0xFF3E0008	See Note	word 2	
0xFF3E00FC	See Note	word 63	

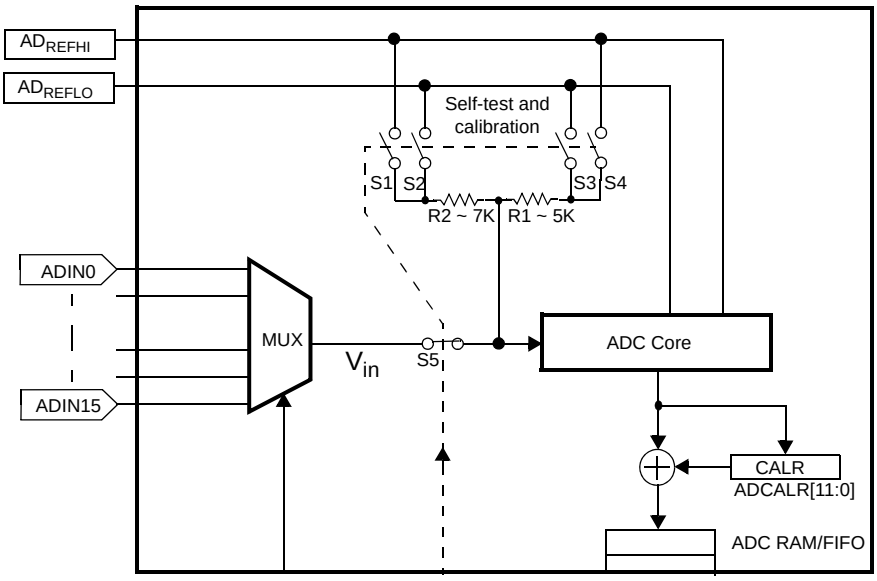
ADC1	31 1	0	
0xFF3E1000	Read 0	P0	ADC RAM Parity
0xFF3E1004	Read 0	P1	
0xFF3E1008	Read 0	P2	
0xFF3E10FC	Read 0	P63	

Note: Bit 31:16 is reserved. Therefore, no parity protection is implemented for bit 31:16.

15.7.2 ADC Self-Test Mode

The self-test mode is used to detect an **open** or a **short** on the ADC input channels. The logic associated with both self-test and calibration is shown in Figure 15-15. Section 15.7.2.1 introduces how to configure the ADC module to do the self-test and Section 15.7.2.2 shows how to use the self-test to detect an open and short on the ADC input channels.

Figure 15-15. Self-Test and Calibration Logic



15.7.2.1 ADC Self-Test Conversions

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. Before a self-test conversion starts, the application must

1. enable the self-test mode by setting the SELF_TEST bit (ADCALCR.24).
2. define the test voltage source by the HILO bit (ADCALCR.8).

Table 15-2 shows how to set the registers to define the test voltage input to the ADC core.

Table 15-2. Self-Test Reference Voltages†

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}

† Switches refer to Figure 15-15.

The conversion group register setting, conversion data storage in self-test mode is the same as a normal conversion. After the test voltage source is defined, the application must follow the instructions in Section 15.3 to start the AD conversion and read the conversion data.

When an AD conversion (either a self-test conversion or a normal AD conversion) is ongoing, the application should NOT:

1. modify the SELF_TEST bit (ADCALCR.24) to enable or disable the self-test mode;
2. modify the HILO bit (ADCALCR.8) to change the test voltage source.

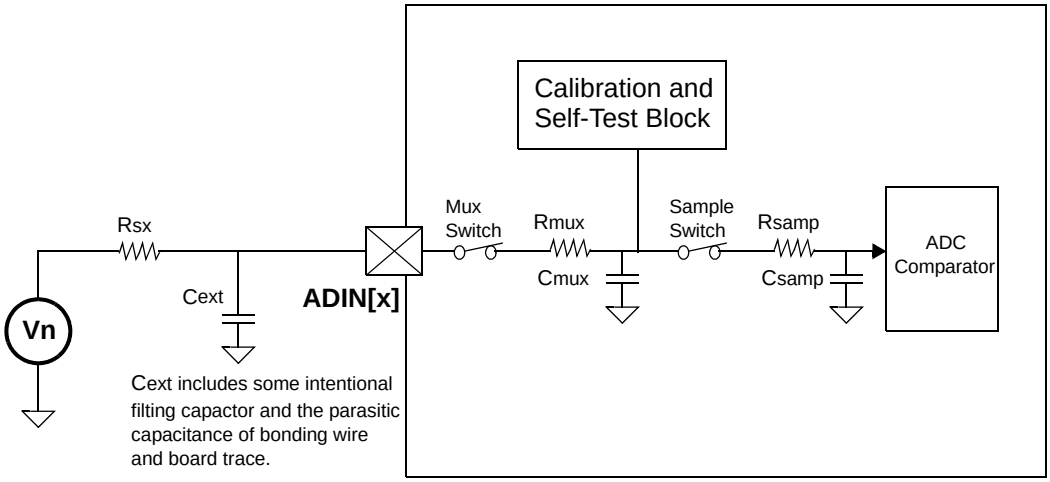
The acquisition time for each conversion is extended to twice the normal configured acquisition time by hardware. The selected reference voltage and the input voltage from the ADINx input channel are both connected to the ADC internal sampling capacitor throughout this extended acquisition period.

Section 15.7.2.2 shows an example to check open and short using ADC self-test conversions.

15.7.2.2 Use of Self-Test Mode to Determine Open/Short on ADC Input Channels

The external circuit shown in Figure 15-16 can be used to check the open and short.

Figure 15-16. Board Level Circuit for Self-Test Mode



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 15-3. 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}
AD_{REFLO} < Vn < AD_{REFHI}	AD_{REFHI}	AD_{REFLO}	Open

15.8 ADC Special Modes

The ADC module supports some special modes for power saving and discharging sampling capacitor purposes.

15.8.1 ADC Powerdown Mode

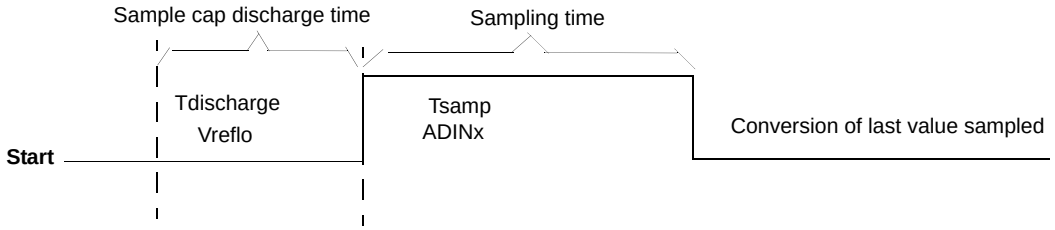
In the power-down mode, the clocks to the ADC module are stopped and the module is in a static state. This results in the lowest possible ADC power consumption. Please refer the device datasheet to identify the power consumption in low power mode.

The ADC enters power-down mode when the appropriate control bit in the power down control register in the peripheral central resource (PCR) registers frame is set. Please refer to the device datasheet to identify the appropriate control bit for powering down the ADC module.

15.8.2 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. This mode can be used if the application requires it.

Figure 15-17. Timing for Sample Capacitor Discharge Mode



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 15-17. 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.

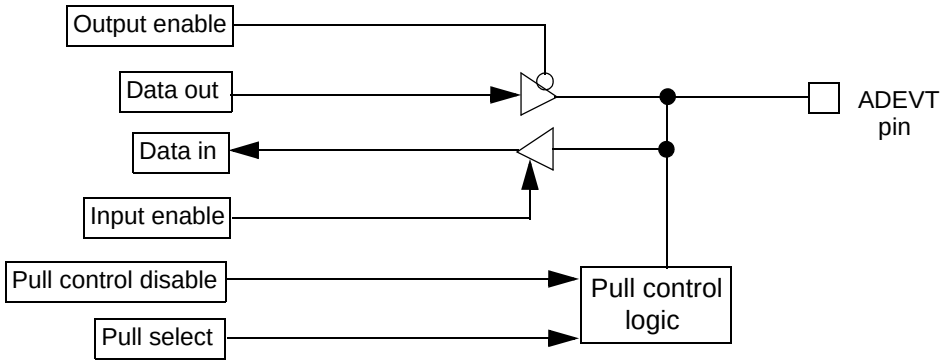
15.9 ADEVT Pin General Purpose I/O Functionality

The ADEVT pin can be configured to be a general-purpose I/O pin. The following sections describe the different ways in which the application can configure the ADEVT pin.

15.9.1 GPIO Functionality

Figure 15-18 illustrates the GPIO functionality.

Figure 15-18. GPIO Functionality



The following apply if the device is out of reset:

- Pull control. The pull control is enabled after reset. 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. The pull control can be disabled by setting the PDIS (pull control disable) bit in the ADEVTPDIS register.

> **Note:**
> The pull function is only available when the pin is configured as input. It is always disabled if the pin is configured as output.

- Input buffer. The input buffer is disabled only if the pin direction is set as input in the ADEVTDIR register AND the pull control is disabled AND pull down is selected. In all other cases, the input buffer is enabled.
- Output buffer. The ADEVT pin can be driven as an output pin if the ADEVTDIR bit is set in the pin direction control register.
- Open-Drain. The open-drain feature can be enabled through ADEVTPDR. This function only applies to output mode.
  - The output buffer is enabled if ADEVTOUT is 0.
  - The output buffer is disabled if ADEVTOUT is 1.

15.9.2 Summary

The behavior of the input buffer, output buffer, and the pull control is summarized in Table 15-4.

Table 15-4. Input Buffer, Output Buffer, and Pull Control Behavior as GPIO Pins

Device under Reset?	Pin Direction (DIR)	Pull Disable (PDIS)	Pull Select (PSEL)	Pull Control	Output Buffer	Input Buffer
Yes	X	X	X	Undefined	Disabled	Undefined
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
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

15.10 ADC Control Registers

All registers in the ADC module are 32-bit, word-aligned; 8-bit, 16-bit and 32-bit accesses are allowed. The following table 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 0xFFF7C000.

Table 15-5. ADC Registers Summary

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
0x00 ADRSTCR Page 781	Reserved															
	Reserved															Reset
0x04 ADOPMODECR Page 782	Reserved							COS	Reserved			Reserved				RAM_TEST_EN
	Reserved							Res	Reserved							ADC_EN
0x08 ADCLOCKCR Page 784	Reserved															
	Reserved											PS[4:0]				
0x0C ADCALCR Page 785	Reserved							SELF_TEST	Reserved							CAL_ST
	Reserved						BRIDGE_EN	HILO	Reserved							CAL_EN
0x10 ADEVMODECR Page 787	Reserved															
	Reserved										EV_CHID	OVR_EV_RAM_IGN	Reserved	EV_8Bit	EV_MODE	FRZ_EV
0x14 ADG1MODECR Page 789	Reserved															
	Reserved										G1_CHID	OVR_G1_RAM_IGN	G1_HW_TRIG	G1_8Bit	G1_MODE	FRZ_G1
0x18 ADG2MODECR Page 791	Reserved															
	Reserved										G2_CHID	OVR_G2_RAM_IGN	G2_HW_TRIG	G2_8Bit	G2_MODE	FRZ_G2
0x1C ADEVSRC Page 794	Reserved															
	Reserved												EV_EDG_SEL	EVSRC[2:0]		

Table 15-5. ADC Registers Summary (Continued)

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
0x20 ADG1SRC Page 794	Reserved															
	Reserved												G1_EDG_SEL	G1SRC[2:0]		
0x24 ADG2SRC Page 794	Reserved															
	Reserved												G2_EDG_SEL	G2SRC[2:0]		
0x28 ADEVINTENA Page 796	Reserved															
	Reserved												EV_END_INT_EN	Res	EV_OVR_INT_EN	EV_THR_INT_EN
0x2C ADG1INTENA Page 797	Reserved															
	Reserved												G1_END_INT_EN	Res	G1_OVR_INT_EN	G1_THR_INT_EN
0x30 ADG2INTENA Page 798	Reserved															
	Reserved												G2_END_INT_EN	Res	G2_OVR_INT_EN	G2_THR_INT_EN
0x34 ADEVINTFLG Page 799	Reserved															
	Reserved												EV_END	EV_MEM_EMPTY	EV_MEM_OVERRUN	EV_THR_INT_FLG
0x38 ADG1INTFLG Page 801	Reserved															
	Reserved												G1_END	G1_MEM_EMPTY	G1_MEM_OVERRUN	G1_THR_INT_FLG
0x3C ADG2INTFLG Page 803	Reserved															
	Reserved												G2_END	G2_MEM_EMPTY	G2_MEM_OVERRUN	G2_THR_INT_FLG
0x40 ADEVINTCR Page 805	Reserved															
	Sign Extension								EV_THR[8:0]							

Table 15-5. ADC Registers Summary (Continued)

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
0x44 ADG1INTCR Page 806	Reserved															
	Sign Extension								G1_THR[8:0]							
0x48 ADG2INTCR Page 807	Reserved															
	Sign Extension								G2_THR[8:0]							
0x4C Reserved	Reserved															
	Reserved															
0x50 Reserved	Reserved															
	Reserved															
0x54 Reserved	Reserved															
	Reserved															
0x58 ADBNDCR Page 808	Reserved							BNDA[8:0]								
	Reserved							BNDB[8:0]								
0x5C ADBNDEND Page 809	Reserved															BUF_Init_Active
	Reserved													BNDEND[2:0]		
0x60 ADEVSAMP Page 811	Reserved															
	Reserved					EV_ACQ[11:0]										

Table 15-5. ADC Registers Summary (Continued)

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
0x64 ADG1SAMP Page 812	Reserved															
	Reserved						G1_ACQ[11:0]									
0x68 ADG2SAMP Page 813	Reserved															
	Reserved						G2_ACQ[11:0]									
0x6C ADEVSR Page 814	Reserved															
	Reserved												EV_MEM_EMPTY	EV_BUSY	EV_STOP	EV_END
0x70 ADG1SR Page 816	Reserved															
	Reserved												G1_MEM_EMPTY	G1_BUSY	G1_STOP	G1_END
0x74 ADG2SR Page 818	Reserved															
	Reserved												G2_MEM_EMPTY	G2_BUSY	G2_STOP	G2_END
0x78 ADEVSEL Page 820	Reserved															
	EV_SEL[15:0]															
0x7C ADG1SEL Page 821	Reserved															
	G1_SEL[15:0]															
0x80 ADG2SEL Page 822	Reserved															
	G2_SEL[15:0]															
0x84 ADCALR Page 823	Reserved															
	Reserved						ADCALR[9:0]									

Table 15-5. ADC Registers Summary (Continued)

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
0x88	Reserved for ADC logic state machine debug purposes															
	Reserved for ADC logic state machine debug purposes															
0x8C ADLASTCONV Page 824	Reserved															
	LAST_CONV[15:0]															
0x90 to 0xAF ADEVBUFFER Page 825	Reserved															
	EV_EMPTY	EV_CHID					EV_DR									
0xB0 to 0xCF ADG1BUFFER Page 826	Reserved															
	G1_EMPTY	G1_CHID					G1_DR									
0xD0 to 0xEF ADG2BUFFER Page 827	Reserved															
	G2_EMPTY	G2_CHID					G2_DR									
0xF0 ADEVEMUBUFFER Page 828	Reserved															
	EV_EMPTY	EV_CHID					EV_DR									
0xF4 ADG1EMUBUFFER Page 828	Reserved															
	G1_EMPTY	G1_CHID					G1_DR									
0xF8 ADG2EMUBUFFER Page 828	Reserved															
	G2_EMPTY	G2_CHID					G2_DR									

Table 15-5. ADC Registers Summary (Continued)

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
0xFC ADEVTDIR Page 829	Reserved															
	Reserved															ADEVT_DIR
0x100 ADEVTOUT Page 830	Reserved															
	Reserved															ADEVT_OUT
0x104 ADEVTIN Page 831	Reserved															
	Reserved															ADEVT_IN
0x108 ADEVTSET Page 832	Reserved															
	Reserved															ADEVT_SET
0x10C ADEVTCLR Page 833	Reserved															
	Reserved															ADEVT_CLR
0x110 ADEVTPDR Page 834	Reserved															
	Reserved															ADEVT_PDR
0x114 ADEVTPDIS Page 835	Reserved															
	Reserved															ADEVT_PDIS
0x118 ADEVTPSEL Page 836	Reserved															
	Reserved															ADEVT_PSEL
0x11C ADEVSAMPDISEN Page 837	Reserved															
	EV_SAMP_DIS_CYC[7:0]								Reserved							EV_SAMP_DIS_EN

Table 15-5. ADC Registers Summary (Continued)

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
0x120 ADG1SAMPDISEN Page 838	Reserved															
	G1_SAMP_DIS_CYC[7:0]								Reserved							G1_SAMP_DIS_EN
0x124 ADG2SAMPDISEN Page 839	Reserved															
	G2_SAMP_DIS_CYC[7:0]								Reserved							G2_SAMP_DIS_EN
0x128 ADMAGINTCR1 Page 840	Reserved	MAG_CHID1[4:0]					MAG_THRESHOLD1[9:0]									
	Reserved			COMP_CHID1[4:0]					Reserved						CHN/THR COMP1	COMP_GE/LT1
0x12C ADMAG1MASK Page 842	Reserved															
	Reserved						MAG_INT1_MASK[9:0]									
0x130 ADMAGINTCR2 Page 840	Reserved	MAG_CHID2[4:0]					MAG_THRESHOLD2[9:0]									
	Reserved			COMP_CHID2[4:0]					Reserved						CHN/THR COMP2	COMP_GE/LT2
0x134 ADMAG2MASK Page 842	Reserved															
	Reserved						MAG_INT2_MASK[9:0]									
0x138 ADMAGINTCR3 Page 840	Reserved	MAG_CHID3[4:0]					MAG_THRESHOLD3[9:0]									
	Reserved			COMP_CHID3[4:0]					Reserved						CHN/THR COMP3	COMP_GE/LT3
0x13C ADMAG3MASK Page 842	Reserved															
	Reserved						MAG_INT3_MASK[9:0]									

Table 15-5. ADC Registers Summary (Continued)

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
0x140 ADMAGINTCR4 Page 840	Reserved	MAG_CHID4[4:0]					MAG_THRESHOLD4[9:0]									
	Reserved			COMP_CHID4[4:0]					Reserved						CHN/THR COMP4	COMP_GE/LT4
0x144 ADMAG4MASK Page 842	Reserved															
	Reserved						MAG_INT4_MASK[9:0]									
0x148 ADMAGINTCR5 Page 840	Reserved	MAG_CHID5[4:0]					MAG_THRESHOLD5[9:0]									
	Reserved			COMP_CHID5[4:0]					Reserved						CHN/THR COMP5	COMP_GE/LT5
0x14C ADMAG5MASK Page 842	Reserved															
	Reserved						MAG_INT5_MASK[9:0]									
0x150 ADMAGINTCR6 Page 840	Reserved	MAG_CHID6[4:0]					MAG_THRESHOLD6[9:0]									
	Reserved			COMP_CHID6[4:0]					Reserved						CHN/THR COMP6	COMP_GE/LT6
0x154 ADMAG6MASK Page 842	Reserved															
	Reserved						MAG_INT6_MASK[9:0]									
0x158 ADMAGINTENA-SET Page 843	Reserved															
	Reserved										MAG_INT_ENA_SET[5:0]					
0x15C ADMAGINTENA-CLR Page 844	Reserved															
	Reserved										MAG_INT_ENA_CLR[5:0]					

Table 15-5. ADC Registers Summary (Continued)

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
0x160 ADMAGTHR INTFLG Page 845	Reserved															
	Reserved										MAG_THR_INT_FLG[5:0]					
0x164 ADMAGTHRINT OFFSET Page 846	Reserved															
	Reserved												MAG_INT_OFF[3:0]			
0x168 ADEVFIFORE-SETCR Page 847	Reserved															
	Reserved															EV_FIFO_RESET
0x16C ADG1FIFORESETCR Page 848	Reserved															
	Reserved															G1_FIFO_RESET
0x170 ADG2FIFORESETCR Page 849	Reserved															
	Reserved															G2_FIFO_RESET
0x174 ADEVRAMADDR Page 850	Reserved															
	Reserved							EV_RAM_ADDR[8:0]								
0x178 ADG1RAMADDR Page 851	Reserved															
	Reserved							G1_RAM_ADDR[8:0]								
0x17C ADG2RAMADDR Page 852	Reserved															
	Reserved							G2_RAM_ADDR[8:0]								
0x180 ADPARCR Page 853	Reserved															
	Reserved							TEST	Reserved				PARITY_ENA[3:0]			

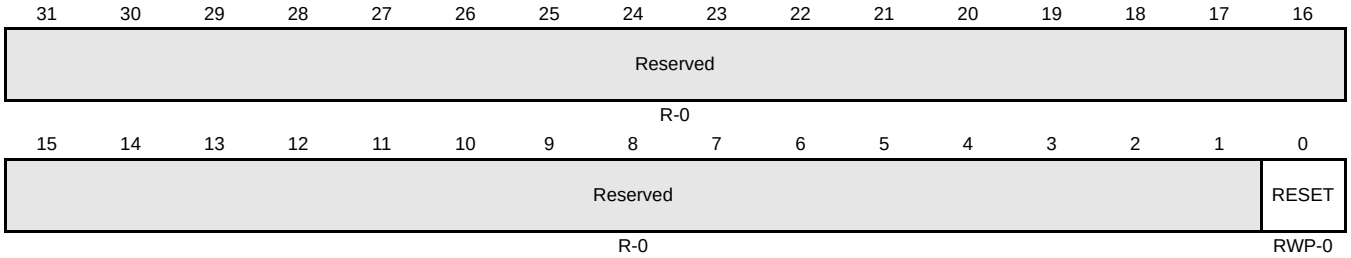
Table 15-5. ADC Registers Summary (Continued)

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
0x184 ADPARADDR Page 854	ERROR_ADDRESS[31:16]															
	ERROR_ADDRESS[15:0]															

15.10.1 ADC Reset Control Register (ADRSTCR)

Figure 15-19 and Table 15-6 describe the ADRSTCR register.

Figure 15-19. ADC Reset Control Register (ADRSTCR) [offset = 0x0]



RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-6. ADC Reset Control Register (ADRSTCR) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved	0	Reads return zeros, writes have no effect.
0	RESET		This bit is used to reset the ADC internal state machines and control/status registers. This reset state is held until this bit is cleared. Read in all modes, write in privileged mode.
		0	Module is released from the reset state.
		1	All the module's internal state machines and the control/status registers are reset.

15.10.2 ADC Operating Mode Control Register (ADOPMODECR)

Figure 15-20 and Table 15-7 describe the ADOPMODECR register.

Figure 15-20. ADC Operating Mode Control Register (ADOPMODECR) [offset = 0x4]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved							COS	Reserved			Reserved				RAM_TEST_EN
R-0							RW-0	R-0			RW-1010				RW-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							Res	Reserved							ADC_EN
R-0							RW-0	R-0							RW-0

RW = Read/Write, RC = Read/Clear, -*n* = value after reset

Table 15-7. ADC Operating Mode Control Register (ADOPMODECR) Field Descriptions

Bit	Name	Value	Description
31–25	Reserved	0	Reads return zeros, writes have no effect.
24	COS		This bit affects *emulation operation only*. It defines whether the ADC core clock (ADCLK) is immediately halted when the emulation system enters debug mode or if it should continue operating normally. **Note**: If COS = 0 when the ADC module 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. User or privileged mode read/write:
		0	ADC module halts all ongoing conversions immediately after emulation mode is entered.
		1	ADC module continues all ongoing conversions as per the configurations of the three conversion groups.
23–21	Reserved	0	Reads return zeros, writes have no effect.
20-17	Reserved	1010	Reserved for TI internal use only. This field shouldn't not be changed to anything else than 1010, otherwise the result of conversion can be unexpected.
16	RAM_TEST_EN		Enable the ADC results FIFO RAM Test Mode. Please refer to Section 15.7.1 for more details. User or privileged mode read/write:
		0	ADC RAM Test Mode is disabled. The application cannot write to the ADC RAM by the CPU or any other master.
		1	ADC RAM Test Mode is enabled. The application can directly write to the ADC RAM by the CPU or any other master.

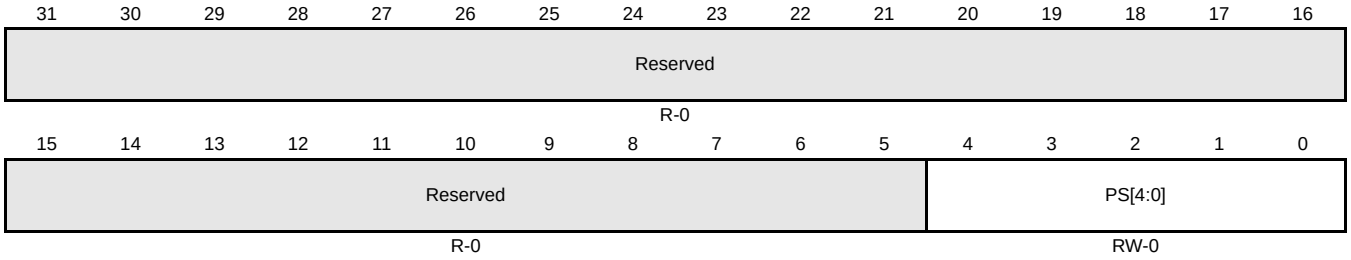
Table 15-7. ADC Operating Mode Control Register (ADOPMODECR) Field Descriptions (Continued)

Bit	Name	Value	Description
15-9	Reserved	0	Reads return zeros, writes have no effect.
8	Reserved	0	Reads return zeros. **Note: Do NOT write a '1' to this bit.**
7-1	Reserved	0	Reads return zeros, writes have no effect.
0	ADC_EN		ADC Enable. This bit must be set to allow the ADC module to be configured to perform any conversions. User or privileged mode read/write:
		0	No ADC conversions can occur. The input channel select registers: ADEVSEL, ADG1SEL, and ADG2SEL are held at their reset values.
		1	ADC conversions can now proceed as configured.

15.10.3 ADC Clock Control Register (ADCLOCKCR)

Figure 15-21 and Table 15-8 describe the ADCLOCKCR register.

Figure 15-21. ADC Clock Control Register (ADCLOCKCR) [offset = 0x8]



RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-8. ADC Clock Control Register (ADCLOCKCR) Field Descriptions

Bit	Name	Value	Description
31–5	Reserved	0	Reads return zeros, writes have no effect.
4-0	PS[4:0]	00000 to 11111	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. User or privileged mode read/write: $t_{C(ADCLK)} = t_{C(VCLK)} * (PS[4:0] + 1)$, where $t_{C(ADCLK)}$ is the period of the ADCLK, and $t_{C(VCLK)}$ is the period of the VCLK.

15.10.4 ADC Calibration Mode Control Register (ADCALCR)

Figure 15-22 and Table 15-9 describe the ADCALCR register.

Figure 15-22. ADC Calibration Mode Control Register (ADCALCR) [offset = 0xC]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved							SELF_TEST	Reserved							CAL_ST
R-0							RW-0	R-0							RS-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						BRIDGE_EN	HILO	Reserved							CAL_EN
R-0						RW-0	RW-0	R--0							RW-0

RW = Read/Write, RC = Read/Clear, RS = Read/Set, -*n* = value after reset

Table 15-9. ADC Calibration Mode Control Register (ADCALCR) Field Descriptions

Bit	Name	Value	Description
31–25	Reserved	0	Reads return zeros, writes have no effect.
24	SELF_TEST		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 HILO bit determines the voltage. The desired conversion mode is configured in the group mode control registers. For more details on the ADC Self Test Mode, please refer to Section 15.7.2. User or privileged mode read/write:
		0	ADC Self Test mode is disabled.
		1	ADC Self Test mode is enabled.
23–17	Reserved	0	Reads return zeros, writes have no effect.
16	CAL_ST		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 read/write:
		0	Read: Calibration conversion has completed, or has not yet been started. Write: Writing 0 to this bit has no effect.
		1	Read: Calibration conversion is in progress. Write: ADC module starts calibration conversion.
15–10	Reserved	0	Reads return zeros, writes have no effect.

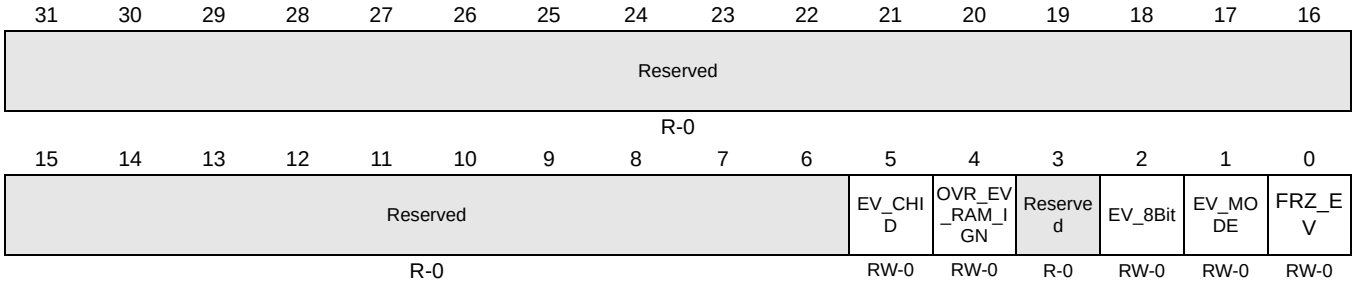
Table 15-9. ADC Calibration Mode Control Register (ADCALCR) Field Descriptions (Continued)

Bit	Name	Value	Description
9	BRIDGE_EN		BRIDGE_ENable.
		0 - 1	In the ADC Calibration Mode, when set with the HILO bit, BRIDGE_EN allows a reference voltage to be converted. The Table 15-1, “Calibration Reference Voltages†,” on page 762 defines the four different reference voltages that can be selected.
			In other modes, this bit has no effect.
8	HILO		ADC self test mode and Calibration Mode Reference Source Selection.
		0 - 1	In the ADC self test mode, this bit defines the test voltage to be combined through a resistor with the selected input pin voltage. The Table 15-2, “Self-Test Reference Voltages†,” on page 766 defines the two different test voltages that can be selected.
			In the ADC Calibration Mode, when set with the BRIDGE_EN bit, this bit defines a reference voltage to be converted. The Table 15-1, “Calibration Reference Voltages†,” on page 762 defines the four different reference voltages that can be selected.
			In the ADC module’s normal operating mode, this bit has no effect.
7–1	Reserved	0	Reads return zeros, writes have no effect.
0	CAL_EN		ADC Calibration Enable. When this bit is set, the analog input channel multiplexer 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.
			Please refer to Section 15.6 for more details on the ADC calibration mode.
			User or privileged mode read/write:
		0	Calibration mode is disabled.
		1	Calibration mode is enabled.

15.10.5 ADC Event Group Operating Mode Control Register (*ADEVMODECR)*

Figure 15-23 and Table 15-10 describe the ADEVMODECR register.

Figure 15-23. ADC Event Group Operating Mode Control Register (ADEVMODECR) [offset = 0x10]



RW = Read/Write, RC = Read/Clear, RS = Read/Set, -*n* = value after reset

Table 15-10. ADC Event Group Operating Mode Control Register (ADEVMODECR) Field Descriptions

Bit	Name	Value	Description
31–6	Reserved	0	Reads return zeros, writes have no effect.
5	EV_CHID		Channel ID Mode for the Event Group. This bit only affects the “read from FIFO” mode. The ADC always stores the channel id in the results FIFO RAM. Any 32-bit read performed in the “read from RAM” mode will return the 5-bit channel id along with the 10-bit conversion result. User or privileged mode read/write: 0 Data is read out of Event Group memory with the CHID field forced to 00000b. 1 Data is read out of Event Group memory with the CHID field containing the ID of the channel to which the digital result belongs.
4	OVR_EV_RAM_IGN		This bit allows the ADC module to overwrite the contents of the Event Group results memory under an overrun condition. User or privileged mode read/write: 0 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. 1 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	0	Reads return zeros, writes have no effect.

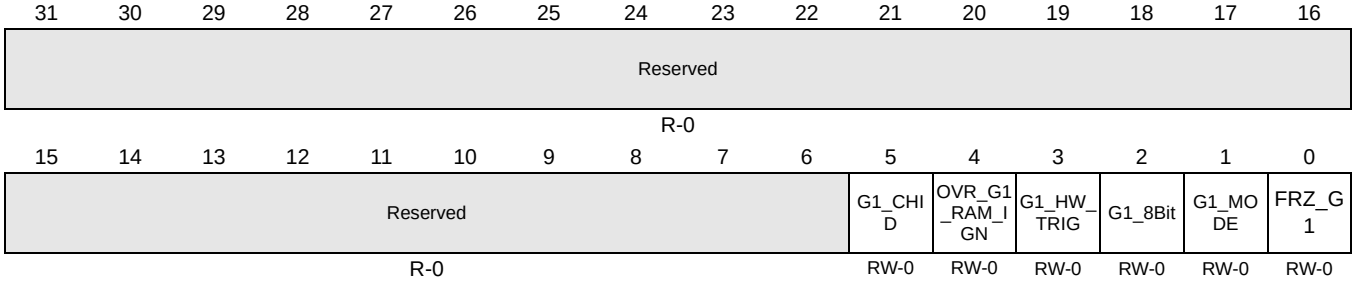
Table 15-10. ADC Event Group Operating Mode Control Register (ADEVMODECR) Field Descriptions

Bit	Name	Value	Description
2	EV_8Bit		Event Group 8-bit result mode. It determines the format in which the Conversion Result is read out when using the FIFO interface of the Event Group results memory, that is, when reading from ADEVBUFFER or ADEVEMUBUFFER.
		0	Conversion Data is read out of the Event Group memory in full 10-bit format.
		1	Conversion Data is read out of the Event Group memory in 8-bit format (RESULT Bits [9:0] >> 2).
1	EV_MODE		This bit selects whether the Event Group, as defined by the ADEVSEL register, is converted in single or continuous conversion mode. User or privileged mode read/write:
		0	Enable single conversion mode of Event Group.
		1	Enable continuous conversion mode of Event Group.
0	FRZ_EV		Event Group Freeze Enable. This bit allows an Event Group conversion sequence to be preempted if a Group1 or a Group2 conversion is requested. After these conversion groups complete their operation, the Event Group resumes its operation from the point where it was preempted. While the Event Group conversion is preempted, 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. User or privileged mode read/write:
		0	Event Group conversions cannot be preempted. All the channels selected for conversion in the Event Group are converted before the ADC can switch over to servicing any other conversion group.
		1	Event Group conversions are preempted whenever there is a request for conversion from Group1 or Group2.

15.10.6 ADC Group1 Operating Mode Control Register (*ADG1MODECR)*

Figure 15-24 and Table 15-11 describe the ADG1MODECR register.

Figure 15-24. ADC Group1 Operating Mode Control Register (ADG1MODECR) [offset = 0x14]



RW = Read/Write, RC = Read/Clear, RS = Read/Set, -*n* = value after reset

Table 15-11. ADC Event Group Operating Mode Control Register (ADG1MODECR) Field Descriptions

Bit	Name	Value	Description
31–6	Reserved	0	Reads return zeros, writes have no effect.
5	G1_CHID		Channel ID Mode for the Group1. This bit only affects the “read from FIFO” mode. The ADC always stores the channel id in the results FIFO RAM. Any 32-bit read performed in the “read from RAM” mode will return the 5-bit channel id along with the 10-bit conversion result. User or privileged mode read/write:
		0	Data is read out of Group1 memory with the CHID field forced to 00000.
		1	Data is read out of Group1 memory with the CHID field containing the ID of the channel to which the digital result belongs.
4	OVR_G1_RAM_IGN		This bit allows the ADC module to overwrite the contents of the Group1 results memory under an overrun condition. User or privileged mode read/write:
		0	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.
		1	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.

Table 15-11. ADC Event Group Operating Mode Control Register (ADG1MODECR) Field Descriptions

Bit	Name	Value	Description
3	G1_HW_TRIG		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, please refer to Section 15.3.6. User or privileged mode read/write:
		0	The Group1 is software-triggered. A Group1 conversion starts whenever the Group1 channel select register (ADG1SEL) is written with a non-zero value.
		1	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		Group1 8-bit result mode. It determines the format in which the Conversion Result is read out when using the FIFO interface of the Group1 results memory, that is, when reading from ADG1BUFFER or ADG1EMUBUFFER.
		0	Conversion Data is read out of the Group1 memory in full 10-bit format.
		1	Conversion Data is read out of the Group1 memory in 8-bit format (RESULT Bits [9:0] >> 2).
1	G1_MODE		This bit selects whether the Group1, as defined by the ADG1SEL register, is converted in single or continuous conversion mode. User or privileged mode read/write:
		0	Enable single conversion mode of Group1.
		1	Enable continuous conversion mode of Group1.
0	FRZ_G1		Group1 Freeze Enable. This bit allows a Group1 conversion sequence to be preempted if an Event Group or a Group2 conversion is requested. After these conversion groups complete their operation, the Group1 resumes its operation from the point where it was preempted. While the Group1 conversion is preempted, 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. User or privileged mode read/write:
		0	Group1 conversions cannot be preempted. All the channels selected for conversion in the Group1 are converted before the ADC can switch over to servicing any other conversion group.
		1	Group1 conversions are preempted whenever there is a request for conversion from Event Group or Group2.

15.10.7 ADC Group2 Operating Mode Control Register (*ADG2MODECR)*

Figure 15-25 and Table 15-12 describe the ADG2MODECR register.

Figure 15-25. ADC Group2 Operating Mode Control Register (ADG2MODECR) [offset = 0x18]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										G2_CHID	OVR_G2_RAM_IGN	G2_HW_TRIG	G2_8Bit	G2_MODE	FRZ_G2
R-0										RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

RW = Read/Write, RC = Read/Clear, RS = Read/Set, -*n* = value after reset

Table 15-12. ADC Group2 Operating Mode Control Register (ADG2MODECR) Field Descriptions

Bit	Name	Value	Description
31–6	Reserved	0	Reads return zeros, writes have no effect.
5	G2_CHID		Channel ID Mode for the Group2. This bit only affects the “read from FIFO” mode. The ADC always stores the channel id in the results FIFO RAM. Any 32-bit read performed in the “read from RAM” mode will return the 5-bit channel id along with the 10-bit conversion result. User or privileged mode read/write:
		0	Data is read out of Group2 memory with the CHID field forced to 00000.
		1	Data is read out of Group2 memory with the CHID field containing the ID of the channel to which the digital result belongs.

Table 15-12. ADC Group2 Operating Mode Control Register (ADG2MODECR) Field Descriptions

Bit	Name	Value	Description
4	OVR_G2_RAM_IGN		This bit allows the ADC module to overwrite the contents of the Group2 results memory under an overrun condition. User or privileged mode read/write:
		0	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.
		1	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		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, please refer to Section 15.3.6. User or privileged mode read/write:
		0	The Group2 is software-triggered. A Group2 conversion starts whenever the Group2 channel select register (ADG2SEL) is written with a non-zero value.
		1	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		Group2 8-bit result mode. It determines the format in which the Conversion Result is read out when using the FIFO interface of the Group2 results memory, that is, when reading from ADG2BUFFER or ADG2EMUBUFFER.
		0	Conversion Data is read out of the Group2 memory in full 10-bit format.
		1	Conversion Data is read out of the Group2 memory in 8-bit format (RESULT Bits [9:0] >> 2).
1	G2_MODE		This bit selects whether the Group2, as defined by the ADG2SEL register, is converted in single or continuous conversion mode. User or privileged mode read/write:
		0	Enable single conversion mode of Group2.
		1	Enable continuous conversion mode of Group2.

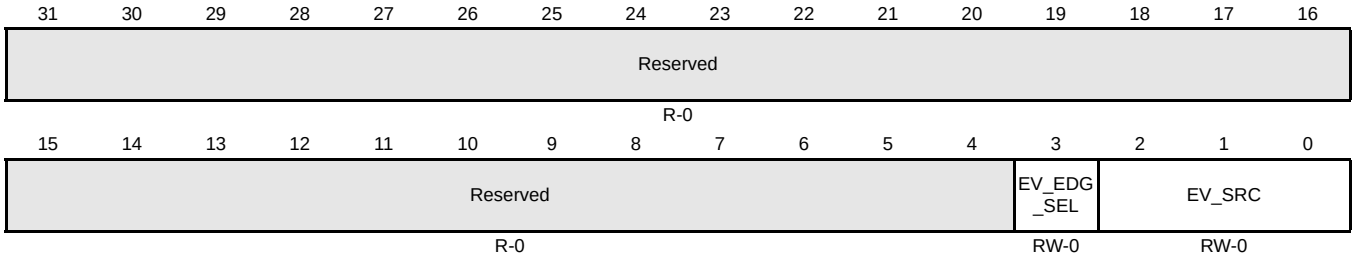
Table 15-12. ADC Group2 Operating Mode Control Register (ADG2MODECR) Field Descriptions

Bit	Name	Value	Description
0	FRZ_G2		Group2 Freeze Enable. This bit allows a Group2 conversion sequence to be preempted if an Event Group or a Group1 conversion is requested. After these conversion groups complete their operation, the Group2 resumes its operation from the point where it was preempted. While the Group2 conversion is preempted, 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. User or privileged mode read/write:
		0	Group2 conversions cannot be preempted. All the channels selected for conversion in the Group2 are converted before the ADC can switch over to servicing any other conversion group.
		1	Group2 conversions are preempted whenever there is a request for conversion from Event Group or Group1.

15.10.8 ADC Trigger Source Select Register (ADEVSRC, ADG1SRC and ADG2SRC)

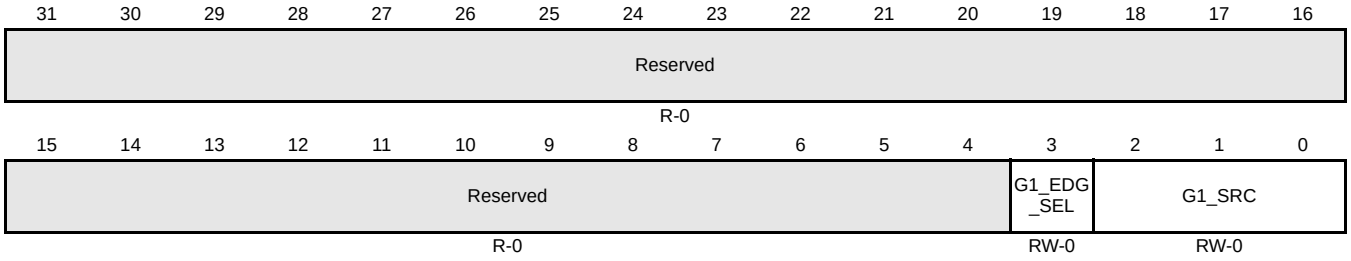
Figure 15-26, Figure 15-27, Figure 15-28and Table 15-13 describe the ADEVSRC, ADG1SRC and ADG2SRC register.

Figure 15-26. ADC Event Group Trigger Source Select Register (ADEVSRC) [offset = 0x1C]



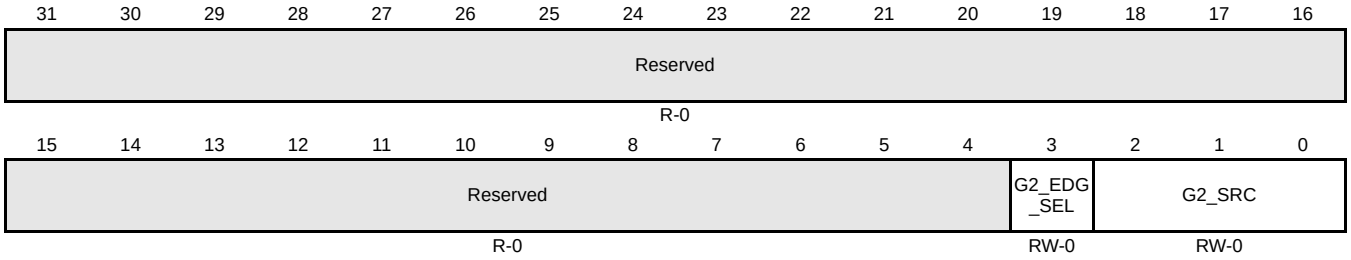
RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Figure 15-27. ADC Group1 Trigger Source Select Register (ADG1SRC) [offset = 0x20]



RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Figure 15-28. ADC Group2 Trigger Source Select Register (ADG2SRC) [offset = 0x24]



RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-13. ADC Trigger Source Select Register (ADEVSRC, ADG1SRC and ADG2SRC) Field Descriptions

Bit	Name	Value	Description
31–4	Reserved	0	Reads return zeros, writes have no effect.

Table 15-13. ADC Trigger Source Select Register (ADEVSRC, ADG1SRC and ADG2SRC) Field Descriptions

Bit	Name	Value	Description
3	EV_EDG_SEL G1_EDG_SEL G2_EDG_SEL		EV Group / Group1/ Group2 Trigger Edge Polarity Select. This bit determines the polarity of the transition on the selected source that triggers the Group conversion. User or privileged mode read/write:
		0	A falling edge on the selected source will trigger the Group conversion.
		1	A rising edge on the selected source will trigger the Group conversion.
2–0	EV_SRC G1_SRC G2_SRC		Event Group / Group1/ Group2 Trigger Source. The application can select different trigger source from ADEVT pin, NHET, RTI and GIOB. User or privileged mode read/write:
		000	ADEVT is the trigger source.
		001	HET[1] is the trigger source.
		010	HET[3] is the trigger source.
		011	HET[16] is the trigger source.
		100	HET[18] is the trigger source.
		101	HET[24] is the trigger source.
		110	HET[26] is the trigger source.
		111	HET[28] is the trigger source.
			Note: the trigger sources listed above are specific to the TMS470M series.

15.10.9 ADC Event Interrupt Enable Control Register (ADEVINTENA)

Figure 15-29 and Table 15-14 describe the ADEVINTENA register.

Figure 15-29. ADC Event Group Interrupt Enable Control Register (ADEVINTENA) [offset = 0x28]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												EV_END_INT_EN	Res	EV_OVR_INT_EN	EV_THR_INT_EN
R-0												RW-0	R-0	RW-0	RW-0

RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-14. ADC Event Group Interrupt Enable Control Register (ADEVINTENA) Field Descriptions

Bit	Name	Value	Description
31–4	Reserved	0	Reads return zeros, writes have no effect.
3	EV_END_INT_EN		Event Group Conversion End Interrupt Enable. Please refer to Section 15.5.1 for more details on the conversion end interrupts. User or privileged mode read/write:
		0	Event Group Conversion End Interrupt is disabled.
		1	Event Group Conversion End Interrupt is Enabled.
2	Reserved	0	Reads return zeros, writes have no effect.
1	EV_OVR_INT_EN		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 please refer to Section 15.5.3 User or privileged mode read/write:
		0	Event Group memory overrun interrupt is disabled.
		1	Event Group memory overrun interrupt is enabled.
0	EV_THR_INT_EN		Event Group Threshold Interrupt Enable. An Event Group threshold interrupt occurs when the programmed Event Group threshold counter counts down to zero. 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. Please refer to Section 15.5.2 for more details on the threshold interrupts. User or privileged mode read/write:
		0	Event Group threshold interrupt is disabled.
		1	Event Group threshold interrupt is enabled.

15.10.10 ADC Group1 Interrupt Enable Control Register (ADG1INTENA)

Figure 15-30 and Table 15-15 describe the ADG1INTENA register.

Figure 15-30. ADC Group1 Interrupt Enable Control Register (ADG1INTENA) [offset = 0x2C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved												G1_END_INT_EN	Res	G1_OVR_INT_EN	G1_THR_INT_EN
R-0												RW-0	R-0	RW-0	RW-0

RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-15. ADC Group1 Interrupt Enable Control Register (ADG1INTENA) Field Descriptions

Bit	Name	Value	Description
31–4	Reserved	0	Reads return zeros, writes have no effect.
3	G1_END_INT_EN		Group1 Conversion End Interrupt Enable. Please refer to Section 15.5.1 for more details on the conversion end interrupts. User or privileged mode read/write:
		0	Group1 Conversion End Interrupt is disabled.
		1	Group1 Conversion End Interrupt is enabled.
2	Reserved	0	Reads return zeros, writes have no effect.
1	G1_OVR_INT_EN		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 please refer to Section 15.5.3 User or privileged mode read/write:
		0	Group1 memory overrun interrupt is disabled.
		1	Group1 memory overrun interrupt is enabled.
0	G1_THR_INT_EN		Group1 Threshold Interrupt Enable. A Group1 threshold interrupt occurs when the programmed Group1 threshold counter counts down to zero. 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. Please refer to Section 15.5.2 for more details on the threshold interrupts. User or privileged mode read/write:
		0	Group1 threshold interrupt is disabled.
		1	Group1 threshold interrupt is enabled.

15.10.11 ADC Group2 Interrupt Enable Control Register (ADG2INTENA)

Figure 15-31 and Table 15-16 describe the ADG1INTENA register.

Figure 15-31. ADC Group2 Interrupt Enable Control Register (ADG2INTENA) [offset = 0x30]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved												G2_END_INT_EN	Res	G2_OVR_INT_EN	G2_THR_INT_EN
R-0												RW-0	R-0	RW-0	RW-0

RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-16. ADC Group2 Interrupt Enable Control Register (ADG2INTENA) Field Descriptions

Bit	Name	Value	Description
31–4	Reserved	0	Reads return zeros, writes have no effect.
3	G2_END_INT_EN		Group2 Conversion End Interrupt Enable. Please refer to Section 15.5.1 for more details on the conversion end interrupts. User or privileged mode read/write:
		0	Group2 Conversion End Interrupt is disabled.
		1	Group2 Conversion End Interrupt is enabled.
2	Reserved	0	Reads return zeros, writes have no effect.
1	G2_OVR_INT_EN		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 please refer to Section 15.5.3 User or privileged mode read/write:
		0	Group2 memory overrun interrupt is disabled.
		1	Group2 memory overrun interrupt is enabled.
0	G2_THR_INT_EN		Group2 Threshold Interrupt Enable. A Group2 threshold interrupt occurs when the programmed Group2 threshold counter counts down to zero. 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. Please refer to Section 15.5.2 for more details on the threshold interrupts. User or privileged mode read/write:
		0	Group2 threshold interrupt is disabled.
		1	Group2 threshold interrupt is enabled.

15.10.12 ADC Event Group Interrupt Flag Register (ADEVINTFLG)

Figure 15-29 and Table 15-17 describe the ADEVINTFLG register.

Figure 15-32. ADC Event Group Interrupt Flag Register (ADEVINTFLG) [offset = 0x34]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												EV_END	EV_MEM_EMPTY	EV_MEM_OVERRUN	EV_THR_INT_FLG
R-0												RC-0	R-1	R-0	RC-0

R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-17. ADC Event Group Interrupt Flag Register (ADEVINTFLG) Field Descriptions

Bit	Name	Value	Description
31–4	Reserved	0	Reads return zeros, writes have no effect.
3	EV_END		Event Group Conversion End. This bit and the Event Group status register bit 0 (ADEVSR.0) are identical. User or privileged mode read: 0 All the channels selected for conversion in the Event Group have not yet been converted. 1 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. This bit can be 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 (ADEVSR.0). - By reading one conversion result from ADEVBUFFER - any Event Group FIFO read - By writing a new set of channels to the Event Group channel select register (ADEVSEL).
2	EV_MEM_EMPTY		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. This bit and the Event Group status register bit 3 (ADEVSR.3) are identical. User or privileged mode read: 0 The Event Group results memory is not empty. 1 The Event Group results memory is empty.

Table 15-17. ADC Event Group Interrupt Flag Register (ADEVINTFLG) Field Descriptions (Continued)

Bit	Name	Value	Description
1	EV_MEM_OVERRUN		Event Group Memory Overrun. This is a read-only bit; writes have no effect. User or privileged mode read:
		0	Event Group results memory has not overrun.
		1	Event Group results memory has overrun.
0	EV_THR_INT_FLG		Event Group Threshold Interrupt Flag. User or privileged mode read:
		0	Event Group buffer has no pending interrupt.
		1	Event Group buffer has pending interrupt due to threshold being reached. This bit can be cleared by writing a ‘1’ ; writing a ‘0’ has no effect.

15.10.13 ADC Group1 Interrupt Flag Register (ADG1INTFLG)

Figure 15-33 and Table 15-18 describe the ADG1INTFLG register.

Figure 15-33. ADC Group1 Interrupt Flag Register (ADG1INTFLG) [offset = 0x38]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												G1_END	G1_MEM_EMPTY	G1_MEM_OVERRUN	G1_THR_INT_FLG
R-0												RC-0	R-1	R-0	RC-0

R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-18. ADC Group1 Interrupt Flag Register (ADG1INTFLG) Field Descriptions

Bit	Name	Value	Description
31–4	Reserved	0	Reads return zeros, writes have no effect.
3	G1_END		Group1 Conversion End. This bit and the Group1 status register bit 0 (ADG1SR.0) are identical. User or privileged mode read:
		0	All the channels selected for conversion in the Group1 have not yet been converted.
		1	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. This bit can be 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 (ADG1SR.0) - By reading one conversion result from ADG1BUFFER - any Group1 FIFO read - By writing a new set of channels to the Group1 channel select register (ADG1SEL).
2	G1_MEM_EMPTY		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. This bit and the Group1 status register bit 3 (ADG1SR.3) are identical. User or privileged mode read:
		0	The Group1 results memory is not empty.
		1	The Group1 results memory is empty.

Table 15-18. ADC Group1 Interrupt Flag Register (ADG1INTFLG) Field Descriptions (Continued)

Bit	Name	Value	Description
1	G1_MEM_OVERRUN		Group1 Memory Overrun. This is a read-only bit; writes have no effect. User or privileged mode read:
		0	Group1 results memory has not overrun.
		1	Group1 results memory has overrun.
0	G1_THR_INT_FLG		Group1 Threshold Interrupt Flag. User or privileged mode read:
		0	Group1 buffer has no pending interrupt.
		1	Group1 buffer has pending interrupt due to threshold being reached. This bit can be cleared only by writing a ‘1’; writing a ‘0’ has no effect.

15.10.14 ADC Group2 Interrupt Flag Register (ADG2INTFLG)

Figure 15-34 and Table 15-19 describe the ADG2INTFLG register.

Figure 15-34. ADC Group2 Interrupt Flag Register (ADG2INTFLG) [offset = 0x3C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												G2_END	G2_MEM_EMPTY	G2_MEM_OVERRUN	G2_THR_INT_FLG
R-0												RC-0	R-1	R-0	RC-0

R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-19. ADC Group2 Interrupt Flag Register (ADG2INTFLG) Field Descriptions

Bit	Name	Value	Description
31–4	Reserved	0	Reads return zeros, writes have no effect.
3	G2_END		Group2 Conversion End. This bit and the Group2 status register bit 0 (ADG2SR.0) are identical. User or privileged mode read:
		0	All the channels selected for conversion in the Group2 have not yet been converted.
		1	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. This bit can be 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 (ADG2SR.0) - By reading one conversion result from ADG2BUFFER - any Group2 FIFO read - By writing a new set of channels to the Group2 channel select register (ADG2SEL).
2	G2_MEM_EMPTY		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. This bit and the Group2 status register bit 3 (ADG2SR.3) are identical. User or privileged mode read:
		0	The Group2 results memory is not empty.
		1	The Group2 results memory is empty.

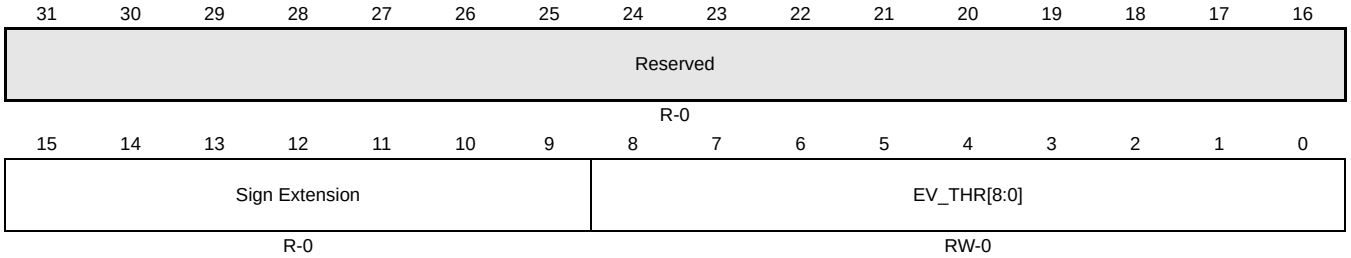
Table 15-19. ADC Group2 Interrupt Flag Register (ADG2INTFLG) Field Descriptions (Continued)

Bit	Name	Value	Description
1	G2_MEM_OVERRUN		Group2 Memory Overrun. This is a read-only bit; writes have no effect. User or privileged mode read:
		0	Group2 results memory has not overrun.
		1	Group2 results memory has overrun.
0	G2_THR_INT_FLG		Group2 Threshold Interrupt Flag. User or privileged mode read:
		0	Group2 buffer has no pending interrupt.
		1	Group2 buffer has pending interrupt due to threshold being reached. This bit can be cleared only by writing a ‘1’; writing a ‘0’ has no effect.

15.10.15 ADC Event Group Threshold Interrupt Control Register (ADEVTHRINTCR)

Figure 15-35 and Table 15-20 describe the ADEVTHRINTCR register.

Figure 15-35. ADC Event Group Threshold Interrupt Control Register (ADEVTHRINTCR) [offset = 0x40]



R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

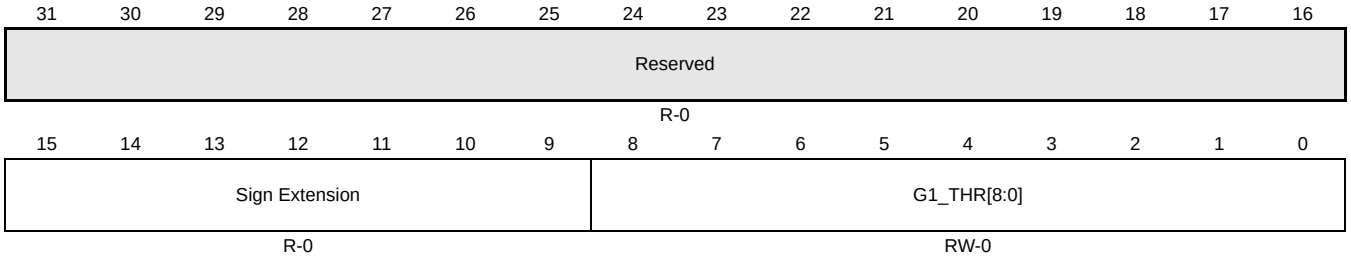
Table 15-20. ADC Event Group Threshold Interrupt Control Register (ADEVTHRINTCR) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved	0	Reads return zeros, writes have no effect.
15–9	Sign Extension		These bits always read the same as the bit 8 of this register, writes have no effect.
8–0	EV_THR		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. After conversions begin, this register tracks the number of ADC sample results currently held in Event Group memory; by decrementing after each new result is placed in the Event Group FIFO by the AD and incrementing as each value is read out of the FIFO by the CPU. When this counter decrements from 1 to 0, it causes the EV_THR_INT_FLG to be set, and an interrupt request is generated if the EV_THR_INT_EN bit is set. Please refer to Section 15.5.2 for more details on the threshold interrupts.

15.10.16 ADC Group1 Threshold Interrupt Control Register (ADG1THRINTCR)

Figure 15-36 and Table 15-21 describe the ADG1THRINTCR register.

Figure 15-36. ADC Group1 Threshold Interrupt Control Register (ADG1THRINTCR) [offset = 0x44]



R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

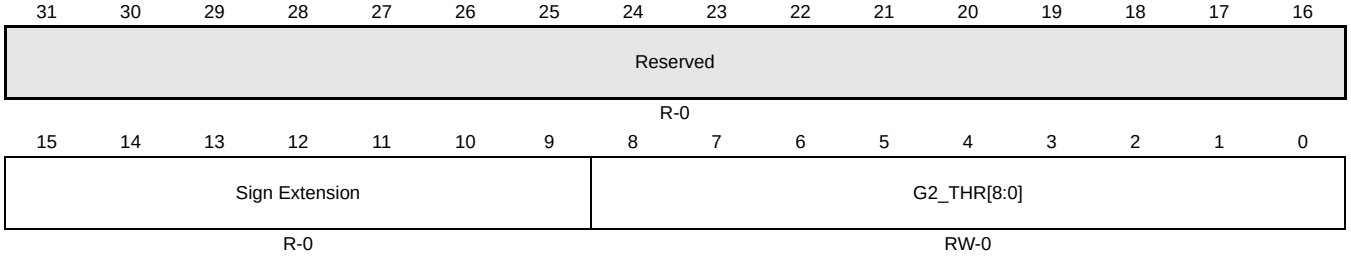
Table 15-21. ADC Group1 Threshold Interrupt Control Register (ADG1THRINTCR) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved	0	Reads return zeros, writes have no effect.
15–9	Sign Extension		These bits always read the same as the bit 8 of this register.
8–0	G1_THR		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. After conversions begin, this register tracks the number of ADC sample results currently held in Group1 memory; by decrementing after each new result is placed in the Group1 FIFO by the AD and incrementing as each value is read out of the FIFO by the CPU. When this counter decrements from 1 to 0, it causes the G1_THR_INT_FLG to be set, and an interrupt request is generated if the G1_THR_INT_EN bit is set. Please refer to Section 15.5.2 for more details on the threshold interrupts.

15.10.17 ADC Group2 Threshold Interrupt Control Register (ADG2THRINTCR)

Figure 15-37 and Table 15-22 describe the ADG2THRINTCR register.

Figure 15-37. ADC Group2 Threshold Interrupt Control Register (ADG2THRINTCR) [offset = 0x48]



R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-22. ADC Group2 Threshold Interrupt Control Register (ADG2THRINTCR) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved	0	Reads return zeros, writes have no effect.
15–9	Sign Extension		These bits always read the same as the bit 8 of this register.
8–0	G2_THR		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. After conversions begin, this register tracks the number of ADC sample results currently held in Group2 memory; by decrementing after each new result is placed in the Group2 FIFO by the AD and incrementing as each value is read out of the FIFO by the CPU. When this counter decrements from 1 to 0, it causes the G2_THR_INT_FLG to be set, and an interrupt request is generated if the G2_THR_INT_EN bit is set. Please refer to Section 15.5.2 for more details on the threshold interrupts.

15.10.18 ADC Results Memory Configuration Register (ADBNDCR)

Figure 15-38 and Table 15-23 describe the ADBNDCR register.

Figure 15-38. ADC Results Memory Configuration Register (ADBNDCR) [offset = 0x58]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved							BNDA[8:0]								
R-0							RW-0								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							BNDB[8:0]								
R-0							RW-0								

R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-23. ADC Results Memory Configuration Register (ADBNDCR) Field Descriptions

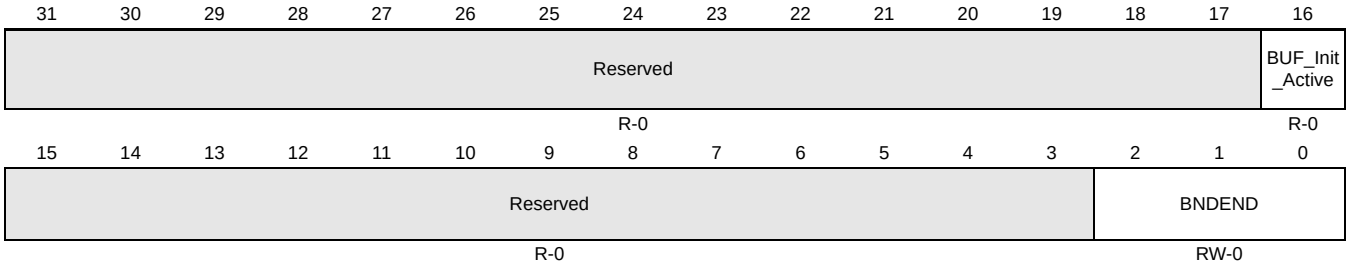
Bit	Name	Value	Description
31–25	Reserved	0	Reads return zeros, writes have no effect.
24–16	BNDA		Buffer Boundary A. These bits determine the memory available for the Event Group conversion results. The memory available is specified in terms of pairs of result buffers. User or privileged mode read/write:
		0	Event Group conversions are not required. If Event Group conversions are performed with the BNDA value of zero, then the Event Group memory size will default 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.
		0x001 to 0x020	A total of (2 * BNDA) buffers are available in the ADC results memory for storing Event Group conversion results.
15–9	Reserved	0	Reads return zeros, writes have no effect.
8–0	BNDB		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 can be determined by subtracting BNDA from BNDB. As a result, BNDB must always be specified as greater than or equal to BNDA. User or privileged mode read:
		0	Event Group as well as Group1 conversions are not required.
		0x001 to 0x020	A total of 2 * (BNDB - BNDA) buffers are available in the ADC results memory for storing Group1 conversion results.

Please refer to Section 15.3.8 for further details on how the conversion results are stored in the ADC results FIFO RAM.

15.10.19 ADC Results Memory Size Configuration Register (ADBNDEND)

Figure 15-39 and Table 15-24 describe the ADBNDCR register.

Figure 15-39. ADC Results Memory Size Configuration Register (ADBNDEND) [offset = 0x5C]



R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-24. ADC Results Memory Size Configuration Register (ADBNDEND) Field Descriptions

Bit	Name	Value	Description
31–17	Reserved	0	Reads return zeros, writes have no effect.
16	BUF_Init_Active		ADC Results Memory Auto-initialization Status. User or privileged mode read only:
		0	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 zeros. For devices requiring parity checking on the ADC results memory, the parity bit in the results memory will also be initialized according to the parity polarity. The parity polarity as well as the auto-initialization process is controlled by the System module. Please refer to the Architecture User Guide for more details.
		1	ADC results memory is being initialized, and the ADC is not available for conversion.
15–3	Reserved	0	Reads return zeros, writes have no effect.

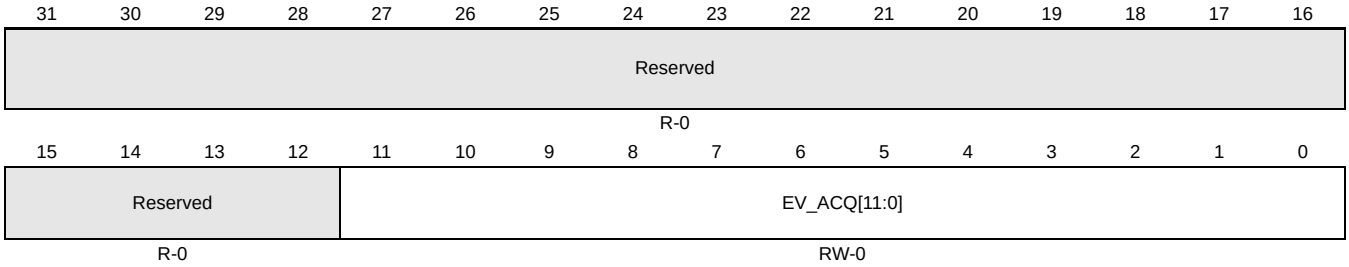
Table 15-24. ADC Results Memory Size Configuration Register (ADBNDEND) Field Descriptions (Continued)

Bit	Name	Value	Description
2–0	BNDEND		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 size of the ADC results memory available on the specific device (64 words available on TMS470M Series). Please refer to the specific device datasheet for information on the memory available for storing the ADC conversion results. For device TMS470M Series, the user must program BNDEND as 010b (64 words) or less. The reason you would do 'less' is for compatibility with another chip that has less ADC RAM. User or privileged mode read/write:
		000b	16 words available for storing ADC conversion results.
		001b	32 words available for storing ADC conversion results.
		010b	64 words available for storing ADC conversion results.
		011b	128 words available for storing ADC conversion results.
		100b	192 words available for storing ADC conversion results.
		101b	256 words available for storing ADC conversion results.
		110b	512 words available for storing ADC conversion results.
		111b	1024 words available for storing ADC conversion results.

15.10.20 ADC Event Group Sampling Time Configuration Register (ADEVSAMP)

Figure 15-40 and Table 15-25 describe the ADEVSAMP register.

Figure 15-40. ADC Event Group Sampling Time Configuration Register (ADEVSAMP) [offset = 0x60]



R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-25. ADC Event Group Sampling Time Configuration Register (ADEVSAMP) Field Descriptions

Bit	Name	Value	Description
31–12	Reserved	0	Reads return zeros, writes have no effect.
11–0	EV_ACQ		Event Group Acquisition Time. These bits define the sampling window (SW) for the Event Group conversions. Sample Window = (EV_ACQ + 2) x $t_{C(ADCLK)}$, where $t_{C(ADCLK)}$ is the ADCLK clock period in ns. There are two factors that determine the minimum sampling window value required: First, the ADC module design requires that Sample Window >= 3 ADCLK cycles. Second, the ADC input impedance necessitates a certain minimum sampling time. This needs to be guaranteed by configuring the EV_ACQ value properly considering the frequency of the ADCLK signal. Please refer to the device datasheet to determine the minimum sampling time for this device.

15.10.21 ADC Group1 Sampling Time Configuration Register (ADG1SAMP)

Figure 15-41 and Table 15-26 describe the ADG1SAMP register.

Figure 15-41. ADC Group1 Sampling Time Configuration Register (ADG1SAMP) [offset = 0x64]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				G1_ACQ[11:0]											
R-0				RW-0											

R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

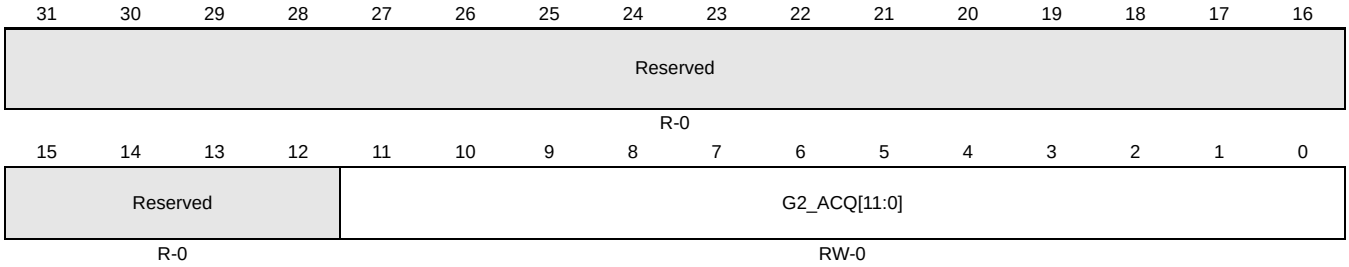
Table 15-26. ADC Group1 Sampling Time Configuration Register (ADG1SAMP) Field Descriptions

Bit	Name	Value	Description
31–12	Reserved	0	Reads return zeros, writes have no effect.
11–0	G1_ACQ		Group1 Acquisition Time. These bits define the sampling window (SW) for the Group1 conversions. Sample Window = (EV_ACQ + 2) x $t_{C(ADCLK)}$. where $t_{C(ADCLK)}$ is the ADCLK clock period in ns. There are two factors that determine the minimum sampling window value required: First, the ADC module design requires that Sample Window >= 3 ADCLK cycles. Second, the ADC input impedance necessitates a certain minimum sampling time. This needs to be guaranteed by configuring the G1_ACQ value properly considering the frequency of the ADCLK signal. Please refer to the device datasheet to determine the minimum sampling time for this device.

15.10.22 ADC Group2 Sampling Time Configuration Register (ADG2SAMP)

Figure 15-42 and Table 15-27 describe the ADG2SAMP register.

Figure 15-42. ADC Group2 Sampling Time Configuration Register (ADG2SAMP) [offset = 0x68]



R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-27. ADC Group2 Sampling Time Configuration Register (ADG2SAMP) Field Descriptions

Bit	Name	Value	Description
31–12	Reserved	0	Reads return zeros, writes have no effect.
11–0	G2_ACQ		Group2 Acquisition Time. These bits define the sampling window (SW) for the Group2 conversions. Sample Window = (EV_ACQ + 2) x $t_{C(ADCLK)}$. where $t_{C(ADCLK)}$ is the ADCLK clock period in ns. There are two factors that determine the minimum sampling window value required: First, the ADC module design requires that Sample Window >= 3 ADCLK cycles. Second, the ADC input impedance necessitates a certain minimum sampling time. This needs to be guaranteed by configuring the G2_ACQ value properly considering the frequency of the ADCLK signal. Please refer to the device datasheet to determine the minimum sampling time for this device.

15.10.23 ADC Event Group Status Register (ADEVSR)

Figure 15-43 and Table 15-28 describe the ADEVSR register.

Figure 15-43. ADC Event Group Status Register (ADEVSR) [offset = 0x6C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												EV_MEM_EMPTY	EV_BUSY	EV_STOP	EV_END
R-0												R-1	R-0	R-0	R-0

R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-28. ADC Event Group Status Register (ADEVSR) Field Descriptions

Bit	Name	Value	Description
31–4	Reserved	0	Reads return zeros, writes have no effect.
3	EV_MEM_EMPTY		Event Group Results Memory Empty. This bit can be effectively used only when the conversion results are read out of the Event Group results memory in the “read from FIFO” mode. This bit and the Event Group Interrupt Flag Register bit 2 (ADEVINTFLG.2) are identical. User or privileged mode read:
		0	The Event Group results memory has unread conversion results.
		1	The Event Group results memory is empty, or all conversion results in the memory have already been read.
2	EV_BUSY		Event Group Conversion Busy. User or privileged mode read:
		0	Event Group conversions are neither in progress nor preempted.
		1	Event Group conversions are either in progress, or are preempted for servicing some other group. This bit will always be set when the Event Group is configured to be in the continuous conversion mode.
1	EV_STOP		Event Group Conversion Stopped. User or privileged mode read:
		0	Event Group conversions are not currently preempted.
		1	Event Group conversions are currently preempted.

Table 15-28. ADC Event Group Status Register (ADEVSR) Field Descriptions (Continued)

Bit	Name	Value	Description
0	EV_END		Event Group Conversions Ended. This bit and the Event Group Interrupt Flag Register bit 3 (ADEVINTFLG.3) are identical. User or privileged mode read:
		0	Event Group conversions have either not been started or have not yet completed since the last time this status bit was cleared.
		1	The conversion for all the channels selected in the Event Group has completed. This bit can be 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

15.10.24 ADC Group1 Status Register (ADG1SR)

Figure 15-44 and Table 15-29 describe the ADG1SR register.

Figure 15-44. ADC Group1 Status Register (ADG1SR) [offset = 0x70]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												G1_MEM_EMPTY	G1_BUSY	G1_STOP	G1_END
R-0												R-1	R-0	R-0	R-0

R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-29. ADC Group1 Status Register (ADG1SR) Field Descriptions

Bit	Name	Value	Description
31–4	Reserved	0	Reads return zeros, writes have no effect.
3	G1_MEM_EMPTY		Group1 Results Memory Empty. This bit can be effectively used only when the conversion results are read out of the Group1 results memory in the “read from FIFO” mode. This bit and the Group1 Interrupt Flag Register bit 2 (ADG1INTFLG.2) are identical. User or privileged mode read:
		0	The Group1 results memory has unread conversion results.
		1	The Group1 results memory is empty, or all conversion results in the memory have already been read.
2	G1_BUSY		Group1 Conversion Busy. User or privileged mode read:
		0	Group1 conversions are neither in progress nor preempted.
		1	Group1 conversions are either in progress, or are preempted for servicing some other group. This bit will always be set when the Group1 is configured to be in the continuous conversion mode.
1	G1_STOP		Group1 Conversion Stopped. User or privileged mode read:
		0	Group1 conversions are not currently preempted.
		1	Group1 conversions are currently preempted.

Bit	Name	Value	Description
0	G1_END		Group1 Conversions Ended. This bit and the Group1 Interrupt Flag Register bit 3 (ADG1INTFLG.3) are identical. User or privileged mode read:
		0	Group1 conversions have either not been started or have not yet completed since the last time this status bit was cleared.
		1	The conversion for all the channels selected in the Group1 has completed. This bit can be 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

15.10.25 ADC Group2 Status Register (ADG2SR)

Figure 15-45 and Table 15-30 describe the ADG2SR register.

Figure 15-45. ADC Group2 Status Register (ADG2SR) [offset = 0x74]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												G2_MEM_EMPTY	G2_BUSY	G2_STOP	G2_END
R-0												R-1	R-0	R-0	R-0

R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-30. ADC Group2 Status Register (ADG2SR) Field Descriptions

Bit	Name	Value	Description
31–4	Reserved	0	Reads return zeros, writes have no effect.
3	G2_MEM_EMPTY		Group2 Results Memory Empty. This bit can be effectively used only when the conversion results are read out of the Group2 results memory in the “read from FIFO” mode. This bit and the Group2 Interrupt Flag Register bit 2 (ADG2INTFLG.2) are identical. User or privileged mode read: 0 The Group2 results memory has unread conversion results. 1 The Group2 results memory is empty, or all conversion results in the memory have already been read.
2	G2_BUSY		Group2 Conversion Busy. User or privileged mode read: 0 Group2 conversions are neither in progress nor preempted. 1 Group2 conversions are either in progress, or are preempted for servicing some other group. This bit will always be set when the Group2 is configured to be in the continuous conversion mode.
1	G2_STOP		Group2 Conversion Stopped. User or privileged mode read: 0 Group2 conversions are not currently preempted. 1 Group2 conversions are currently preempted.

Table 15-30. ADC Group2 Status Register (ADG2SR) Field Descriptions (Continued)

Bit	Name	Value	Description
0	G2_END		Group2 Conversions Ended. This bit and the Group2 Interrupt Flag Register bit 3 (ADG2INTFLG.3) are identical. User or privileged mode read:
		0	Group2 conversions have either not been started or have not yet completed since the last time this status bit was cleared.
		1	The conversion for all the channels selected in the Group2 has completed. This bit can be 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

15.10.26 ADC Event Group Channel Select Register (ADEVSEL)

Figure 15-46 and Table 15-31 describe the ADEVSEL register.

Figure 15-46. ADC Event Group Channel Select Register (ADEVSEL) [offset = 0x78]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
RW-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EV_SEL[15:0]															
RW-0															

R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-31. ADC Event Group Channel Select Register (ADEVSEL) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved	0	Reads return zeros, writes have no effect.
15–0	EV_SEL		Event Group channels selected. User or privileged mode read/write:
		0	No ADC input channel is selected for conversion in the Event Group.
		Non-zero	Setting bit EV_SEL[x] means that pin ADIN[x] (x=0~7) or ADSIN[x] (x=8~15) will be converted as part of the Event Group. All the selected channels will be converted in ascending order when the Event Group is triggered.

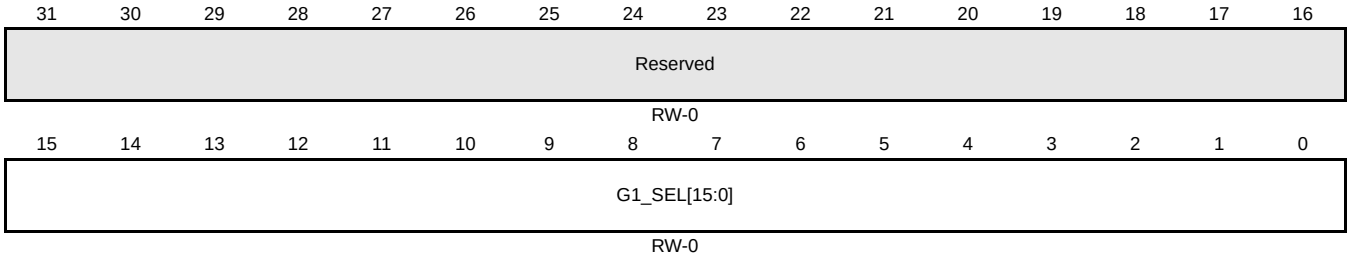
Note: Clearing ADEVSEL During a Conversion
Writing 0x0000 to ADEVSEL stops the Event Group conversions. This does not cause the ADC Event Group results Memory pointer or the Event Group Threshold Register to be reset.

Note: 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 again so that correct number of conversions happen before a Threshold interrupt request is generated.

15.10.27 ADC Group1 Channel Select Register (ADG1SEL)

Figure 15-47 and Table 15-32 describe the ADG1SEL register.

Figure 15-47. ADC Group1 Channel Select Register (ADG1SEL) [offset = 0x7C]



R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-32. ADC Group1 Channel Select Register (ADG1SEL) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved	0	Reads return zeros, writes have no effect.
15–0	G1_SEL		Group1 channels selected. User or privileged mode read/write:
		0	No ADC input channel is selected for conversion in the Group1.
		Non-zero	Setting bit G1_SEL[x] means that pin ADIN[x] (x=0~7) or ADSIN[x] (x=8~15) will be converted as part of the Event Group. All the selected channels will be converted in ascending order when the Group1 is triggered.

Note: Clearing ADG1SEL During a Conversion
Writing 0x0000 to ADG1SEL stops the Group1 conversions. This does not cause the ADC Group1 Results Memory pointer or the Group1 Threshold Register to be reset.

Note: 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 again so that correct number of conversions happen before a Threshold interrupt request is generated.

15.10.28 ADC Group2 Channel Select Register (ADG2SEL)

Figure 15-48 and Table 15-33 describe the ADG2SEL register.

Figure 15-48. ADC Group2 Channel Select Register (ADG2SEL) [offset = 0x80]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
G2_SEL[15:0]															
RW-0															

R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-33. ADC Group2 Channel Select Register (ADG2SEL) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved	0	Reads return zeros, writes have no effect.
15–0	G2_SEL		Group2 channels selected. User or privileged mode read/write:
		0	No ADC input channel is selected for conversion in the Group2.
		Non-zero	Setting bit G2_SEL[x] means that pin ADIN[x] (x=0~7) or ADSIN[x] (x=8~15) will be converted as part of the Event Group. All the selected channels will be converted in ascending order when the Group2 is triggered.

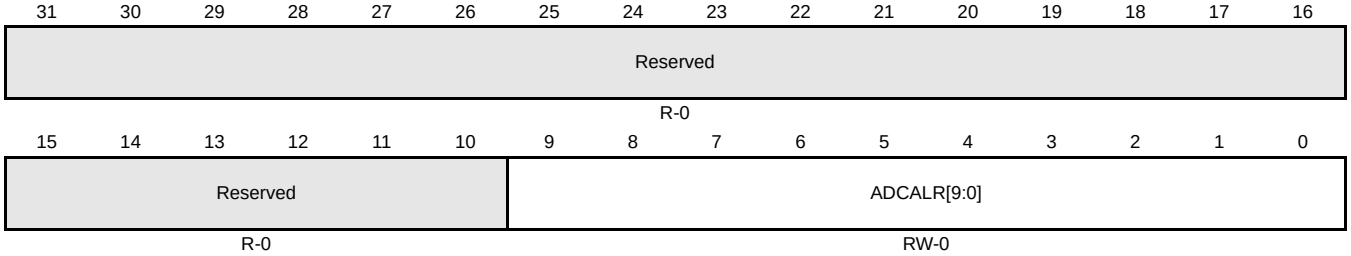
Note: Clearing ADG2SEL During a Conversion
Writing 0x0000 to ADG2SEL stops the Group2 conversions. This does not cause the ADC Group2 Results Memory pointer or the Group2 Threshold Register to be reset.

Note: 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 again so that correct number of conversions happen before a Threshold interrupt request is generated.

15.10.29 ADC Calibration and Error Offset Correction Register (ADCALR)

Figure 15-49 and Table 15-34 describe the ADCALR register.

Figure 15-49. ADC Calibration and Error Offset Correction Register (ADCALR) [offset = 0x84]



R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-34. ADC Calibration and Error Offset Correction Register (ADCALR) Field Descriptions

Bit	Name	Value	Description
31–10	Reserved	0	Reads return zeros, writes have no effect.
9–0	ADCALR		ADC Calibration Result and Offset Error Correction Value. 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 can then compute the correction for the offset error and this correction value needs to be written back to the ADCALR register in the 10-bit 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, please refer to the Section 15.6.

15.10.30 ADC Channel Last Conversion Value Register (ADLASTCONV)

Figure 15-50 and Table 15-35 describe the ADLASTCONV register.

Figure 15-50. ADC Channel Last Conversion Value Register (ADLASTCONV) [offset = 0x8C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-U															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LAST_CONV[15:0]															
R-U															

R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

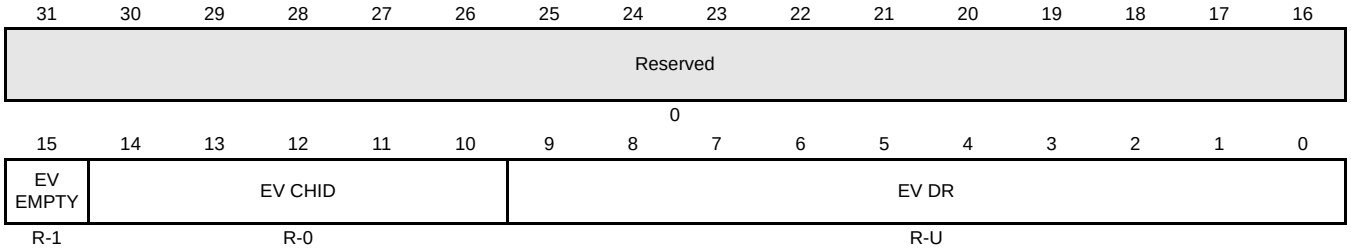
Table 15-35. ADC Channel Last Conversion Value Register (ADLASTCONV) Field Descriptions

Bit	Name	Value	Description
31–0	LAST_CONV		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. Any conversion result from 0 to 0x1FF resoves as 0, otherwise resolves as 1. In other words, this register acts as a digital input register and can be 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:
		0	A level lower than the midpoint reference voltage was measured at the last conversion for this channel
		1	A level higher than the midpoint reference voltage was measured at the last conversion for this channel.

15.10.31 ADC Event Group Results FIFO (ADEVBUFFER)

Figure 15-51 and Table 15-36 describe the ADEVBUFFER register.

Figure 15-51. ADC Event Group Results FIFO (ADEVBUFFER) [offset = 0x90 - 0xAF]



R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

The Event Group results FIFO location is aliased eight times, so that any 32-bit word-aligned read from the address range 0x90 to 0xAF 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.

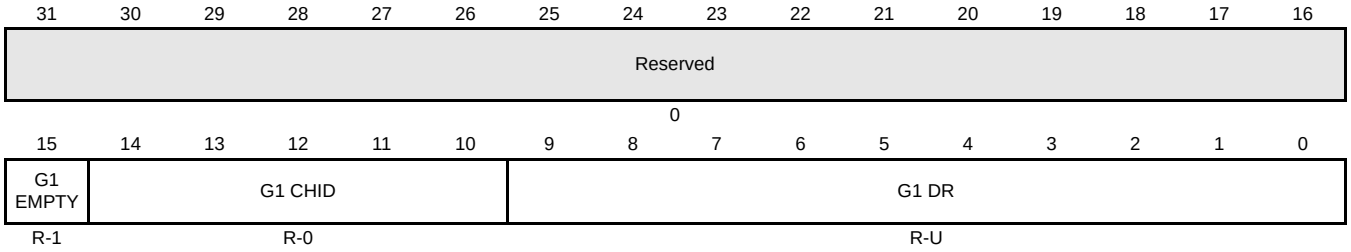
Table 15-36. ADC Event Group Results FIFO (ADEVBUFFER) Field Descriptions

Bit	Name	Value	Description
30–16	Reserved	0	Reads return zeros, writes have no effect.
15	EV_EMPTY		Event Group FIFO Empty. This bit is applicable only when the “read from FIFO” mode is used for reading the Event Group conversion results. User or privileged mode read:
		0	The data in the EV_DR field of this buffer is valid.
		1	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.
14–10	EV_CHID		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. User or privileged mode read:
		0	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).
		00001b to 11111b	The conversion result in the EV_DR field of this buffer is from the ADC input channel number denoted by the EV_CHID field (e.g. 00001b means the conversion result is from the ADC input channel 1).
9–0	EV_DR		Event Group Digital Conversion Result.

15.10.32 ADC Group1 Results FIFO (ADG1BUFFER)

Figure 15-52 and Table 15-37 describe the ADG1BUFFER register.

Figure 15-52. ADC Group1 Results FIFO (ADG1BUFFER) [offset = 0xB0 - 0xCF]



R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

The Group1 results FIFO location is aliased eight times, so that any 32-bit word-aligned read from the address range 0xB0 to 0xCF 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.

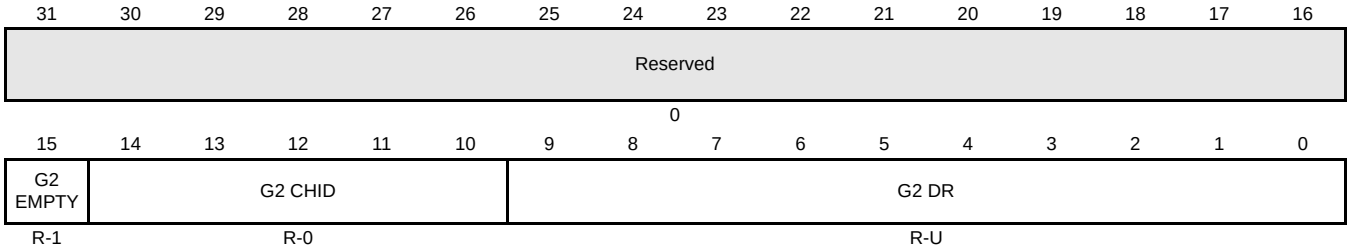
Table 15-37. ADC Group1 Results FIFO (ADG1BUFFER) Field Descriptions

Bit	Name	Value	Description
30–16	Reserved	0	Reads return zeros, writes have no effect.
15	G1_EMPTY		Group1 FIFO Empty. This bit is applicable only when the “read from FIFO” mode is used for reading the Group1 conversion results. User or privileged mode read:
		0	The data in the G1_DR field of this buffer is valid.
		1	The data in the G1_DR field of this buffer is not valid and there are no valid data in the Group1 results memory.
14–10	G1_CHID		Group1 Channel Id. These bits are also applicable only when the “read from FIFO” mode is used for reading the Group1 conversion results. User or privileged mode read:
		0	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).
		00001b to 11111b	The conversion result in the G1_DR field of this buffer is from the ADC input channel number denoted by the G1_CHID field (e.g. 00001b means the conversion result is from the ADC input channel 1).
9–0	G1_DR		Group1 Digital Conversion Result.

15.10.33 ADC Group2 Results FIFO (ADG2BUFFER)

Figure 15-52 and Table 15-37 describe the ADG2BUFFER register.

Figure 15-53. ADC Group2 Results FIFO (ADG2BUFFER) [offset = 0xD0 - 0xEF]



R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

The Group2 results FIFO location is aliased eight times, so that any 32-bit word-aligned read from the address range 0xD0 to 0xEF 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.

Table 15-38. ADC Group2 Results FIFO (ADG2BUFFER) Field Descriptions

Bit	Name	Value	Description
30–16	Reserved	0	Reads return zeros, writes have no effect.
15	G2_EMPTY		Group2 FIFO Empty. This bit is applicable only when the “read from FIFO” mode is used for reading the Group2 conversion results. User or privileged mode read:
		0	The data in the G2_DR field of this buffer is valid.
		1	The data in the G2_DR field of this buffer is not valid and there are no valid data in the Group1 results memory.
14–10	G2_CHID		Group2 Channel Id. These bits are also applicable only when the “read from FIFO” mode is used for reading the Group2 conversion results. User or privileged mode read:
		0	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).
		00001b to 11111b	The conversion result in the G2_DR field of this buffer is from the ADC input channel number denoted by the G2_CHID field (e.g. 00001b means the conversion result is from the ADC input channel 1).
9–0	G2_DR		Group2 Digital Conversion Result.

15.10.34 ADC Event Group Results Emulation FIFO (ADEVEMUBUFFER) [offset = 0xF0]

Reading this register returns an identical result as reading the ADEVBUFFER but that it doesn't affect the status flags in the Event Group interrupt flag register or the Event Group status register, and the FIFO threshold counter. Please refer to the ADEVBUFFER register for bit field information.

15.10.35 ADC Group1 Results Emulation FIFO (ADG1EMUBUFFER) [offset = 0xF4]

Reading this register returns an identical result as reading the ADG1BUFFER but that it doesn't affect the status flags in the Group1 interrupt flag register or the Group1 status register, and the FIFO threshold counter. Please refer to the ADG1BUFFER register for bit field information.

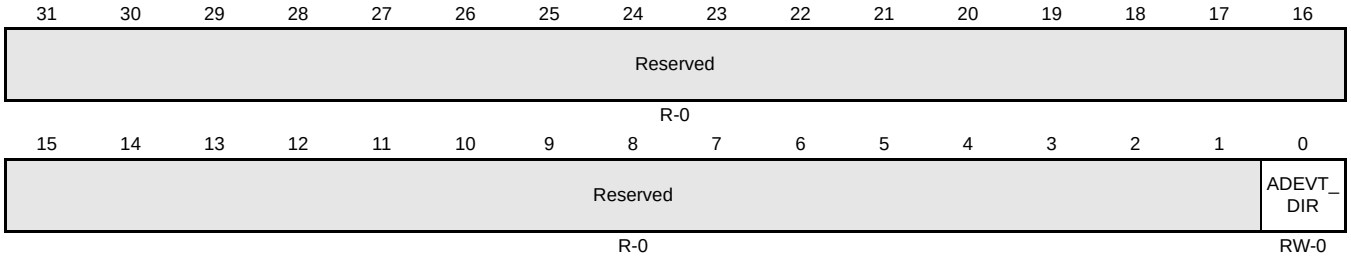
15.10.36 ADC Group2 Results Emulation FIFO (ADG2EMUBUFFER) [offset = 0xF8]

Reading this register returns an identical result as reading the ADG2BUFFER but that it doesn't affect the status flags in the Group2 interrupt flag register or the Group2 status register, and the FIFO threshold counter. Please refer to the ADG2BUFFER register for bit field information.

15.10.37 ADC ADEVT Pin Direction Control Register (ADEVTDIR)

Figure 15-54 and Table 15-39 describe the ADEVTDIR register.

Figure 15-54. ADC ADEVT Pin Direction Control Register (ADEVTDIR) [offset = 0xFC]



R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

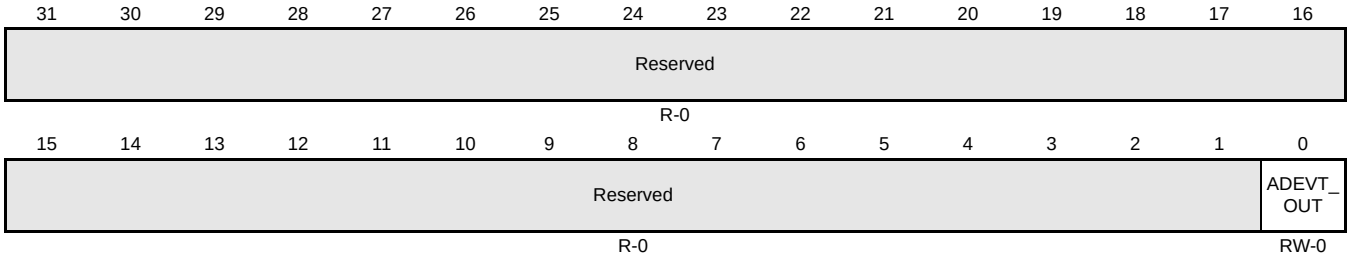
Table 15-39. ADC ADEVT Pin Direction Control Register (ADEVTDIR) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved	0	Reads return zeros, writes have no effect.
0	ADEVT_DIR		ADEVT Pin Direction. Any operating mode read/write:
		0	The ADEVT pin is an input. **Note: If the pin direction is set as an input , the output buffer is tristated.**
		1	The ADEVT pin is an output. **Note: The input buffer is always enabled except for the situation stated in Section 15.9.2.**

15.10.38 ADC ADEVT Pin Output Value Control Register (ADEVTOUT)

Figure 15-55 and Table 15-40 describe the ADEVTOUT register.

Figure 15-55. ADC ADEVT Pin Output Value Control Register (ADEVTOUT) [offset = 0x100]



R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

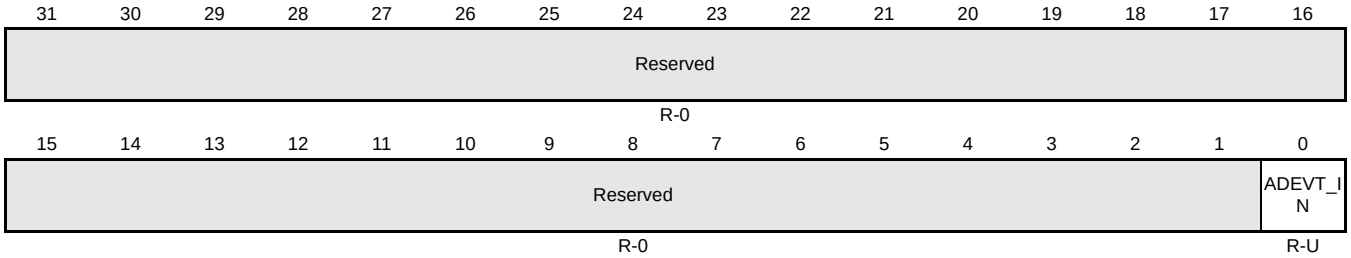
Table 15-40. ADC ADEVT Pin Output Value Control Register (ADEVTOUT) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved	0	Reads return zeros, writes have no effect.
0	ADEVT_OUT		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 operating mode read/write: 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 ADEVTPDR bit = 1 and ADEVTOUT bit = 1.** **Note: ADEVT pin is placed in output mode by setting the ADEVTDIR bit to 1.**

15.10.39 ADC ADEVT Pin Input Value Register (ADEVTIN)

Figure 15-56 and Table 15-41 describe the ADEVTOUT register.

Figure 15-56. ADC ADEVT Pin Input Value Register (ADEVTIN) [offset = 0x104]



R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-41. ADC ADEVT Pin Input Value Register (ADEVTIN) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved	0	Reads return zeros, writes have no effect.
0	ADEVT_IN		ADEVT Pin Input Value. This is a read-only bit which reflects the logic level on the ADEVT pin.
			Any operating mode read:
		0	The pin is at logic low (0).
		1	The pin is at logic high (1).

15.10.40 ADC ADEVT Pin Set Register (ADEVTSET)

Figure 15-57 and Table 15-42 describe the ADEVTSET register.

Figure 15-57. ADC ADEVT Pin Set Register (ADEVTSET) [offset = 0x108]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															ADEVT_SET
R-0															RW-0

R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

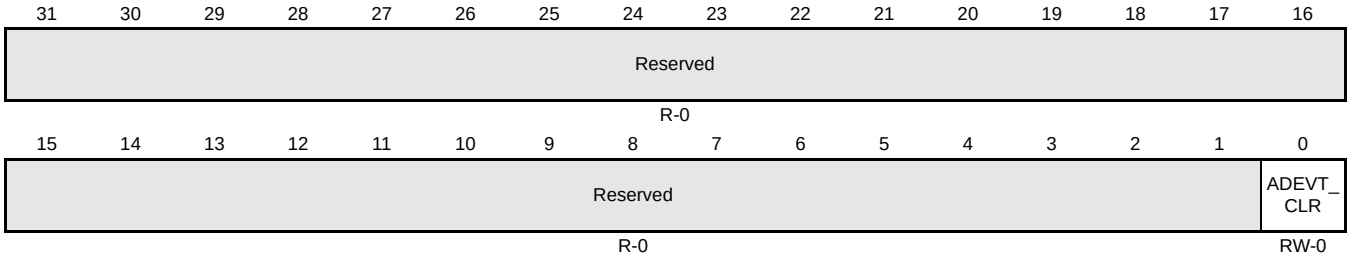
Table 15-42. ADC ADEVT Pin Set Register (ADEVTSET) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved	0	Reads return zeros, writes have no effect.
0	ADEVT_SET		ADEVT Pin Set. This bit drives the output of the ADEVT pin high. Any operating mode read/write:
		0	*Write:* Writing a zero has no effect.
		1	*Write:* ADEVTOUT[0] is driven to logic high.
			Note: The current logic state of the ADEVTOUT[0] bit will also be displayed by this bit. **Note: ADEVT pin is placed in output mode by setting the ADEVTDIR bit to 1.**

15.10.41 ADC ADEVT Pin Clear Register (ADEVTCLR)

Figure 15-58 and Table 15-43 describe the ADEVTCLR register.

Figure 15-58. ADC ADEVT Pin Clear Register (ADEVTCLR) [offset = 0x10C]



R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

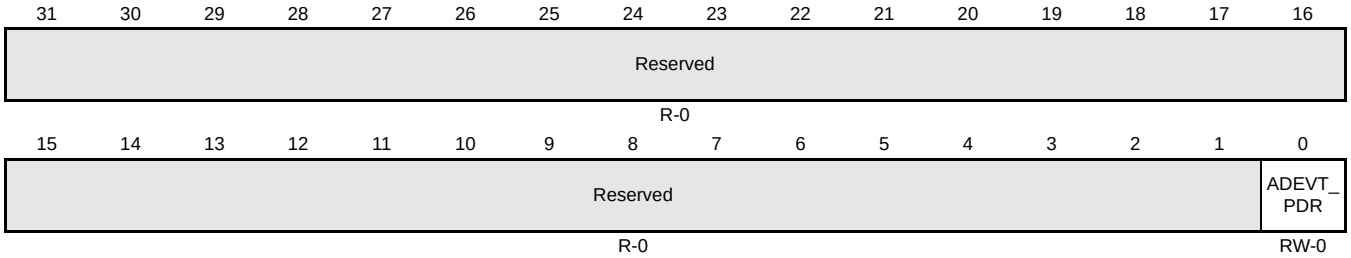
Table 15-43. ADC ADEVT Pin Clear Register (ADEVTCLR) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved	0	Reads return zeros, writes have no effect.
0	ADEVT_CLR		ADEVT Pin Clear. This bit drives the output of the ADEVT pin low. Any operating mode read/write:
		0	*Write:* Writing a zero has no effect.
		1	*Write:* ADEVTOUT[0] is driven to logic low.
			Note: The current logic state of the ADEVTOUT[0] bit will also be displayed by this bit. **Note: ADEVT pin is placed in output mode by setting the ADEVTDIR bit to 1.**

15.10.42 ADC ADEVT Pin Open Drain Enable Register (ADEVTPDR)

Figure 15-59 and Table 15-44 describe the ADEVTPDR register.

Figure 15-59. ADC ADEVT Pin Open Drain Enable Register (ADEVTPDR) [offset = 0x110]



R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

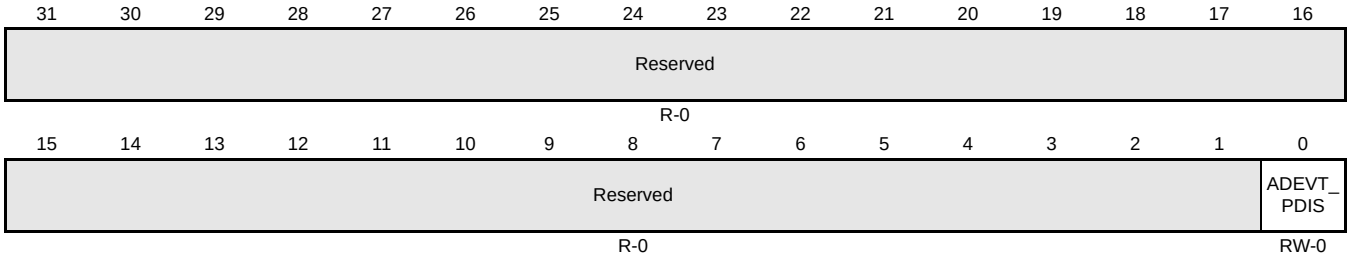
Table 15-44. ADC ADEVT Pin Open Drain Enable Register (ADEVTPDR) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved	0	Reads return zeros, writes have no effect.
0	ADEVT_PDR		ADEVT Pin Open Drain Enable. Any operating mode read/write:
		0	The ADEVT pin is configured in push/pull (normal GIO) mode. The output voltage is driven to V_{OL} or lower if ADEVTOUT[0] =0 and V_{OH} or higher if ADEVTOUT[0] bit =1.
		1	The ADEVT pin is configured in open drain mode. The ADEVTOUT[0] bit controls the state of the ADEVT output buffer: ADEVTOUT[0] = 0 The ADEVT output buffer is driven low; ADEVTOUT[0] = 1 The ADEVT output buffer is tristated.

15.10.43 ADC ADEVT Pin Pull Control Disable Register (ADEVTPDIS)

Figure 15-60 and Table 15-45 describe the ADEVTPDIS register.

Figure 15-60. ADC ADEVT Pin Pull Control Disable Register (ADEVTPDIS) [offset = 0x114]



R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

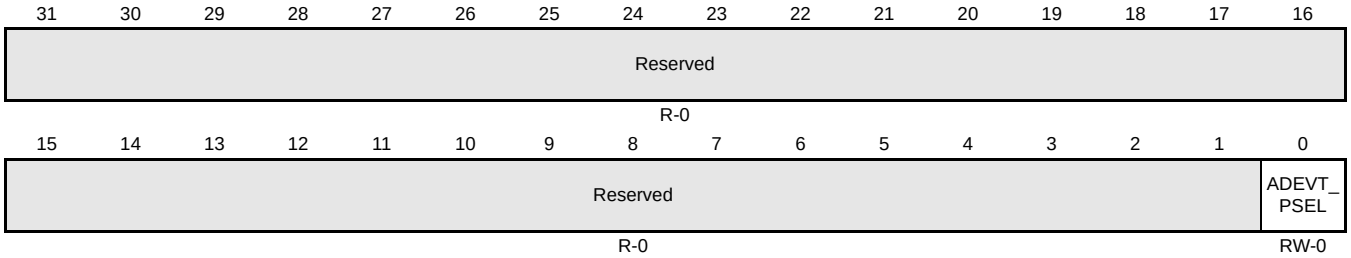
Table 15-45. ADC ADEVT Pin Pull Control Disable Register (ADEVTPDIS) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved	0	Reads return zeros, writes have no effect.
0	ADEVT_PDIS		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 operating mode read/write:
		0	The pull functionality is enabled.
		1	The pull functionality is disabled.

15.10.44 ADC ADEVT Pin Pull Control Select Register (ADEVTPSEL)

Figure 15-61 and Table 15-46 describe the ADEVTPSEL register.

Figure 15-61. ADC ADEVT Pin Pull Control Select Register (ADEVTPSEL) [offset = 0x118]



R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

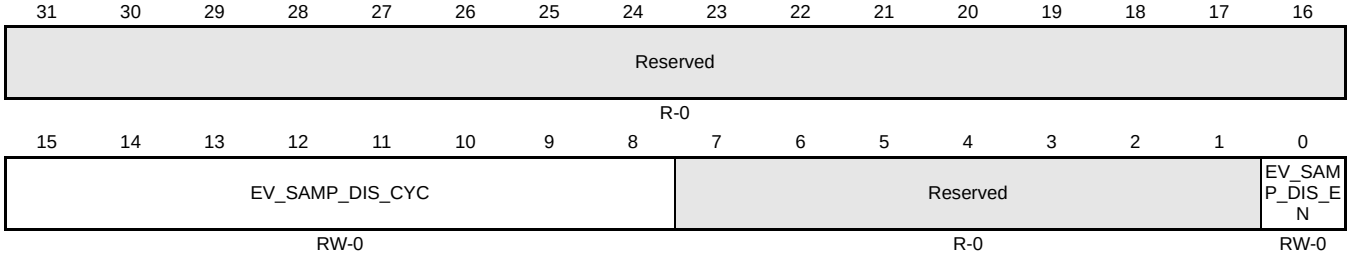
Table 15-46. ADC ADEVT Pin Pull Control Select Register (ADEVTPSEL) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved	0	Reads return zeros, writes have no effect.
0	ADEVT_PSEL		ADEVT Pin Pull Control Select. Any operating mode read/write:
		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 ADEVTPDIS to 0.

15.10.45 ADC Event Group Sample Cap Discharge Control Register (ADEVSAMPDISEN)

Figure 15-62 and Table 15-47 describe the ADEVSAMPDISEN register.

Figure 15-62. ADC Event Group Sample Cap Discharge Control Register (ADEVSAMPDISEN) [offset = 0x11C]



R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-47. ADC Event Group Sample Cap Discharge Control Register (ADEVSAMPDISEN) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved	0	Reads return zeros, writes have no effect.
15–8	EV_SAMP_DIS_CYC		Event Group sample cap discharge cycles. These bits specify the ADC internal sampling capacitor discharge duration before sampling the input channel voltage: (EV_SAMP_DIS_CYC) * $t_{C(ADCLK)}$. where $t_{C(ADCLK)}$ is the ADCLK clock period in ns.
7–1	Reserved	0	Reads return zeros, writes have no effect.
0	EV_SAMP_DIS_EN		Event Group sample cap discharge enable. User or privileged mode read/write:
		0	Event Group sample cap discharge mode is disabled.
		1	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.

15.10.46 ADC Group1 Sample Cap Discharge Control Register (ADG1SAMPDISEN)

Figure 15-63 and Table 15-48 describe the ADG1SAMPDISEN register.

Figure 15-63. ADC Group1 Sample Cap Discharge Control Register (ADG1SAMPDISEN) [offset = 0x120]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
G1_SAMP_DIS_CYC								Reserved							G1_SAMP_DIS_EN
RW-0								R-0							RW-0

R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-48. ADC Group1 Sample Cap Discharge Control Register (ADG1SAMPDISEN) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved	0	Reads return zeros, writes have no effect.
15–8	G1_SAMP_DIS_CYC		Group1 sample cap discharge cycles. These bits specify the ADC internal sampling capacitor discharge duration before sampling the input channel voltage: (G1_SAMP_DIS_CYC) * $t_{C(ADCLK)}$. where $t_{C(ADCLK)}$ is the ADCLK clock period in ns.
7–1	Reserved	0	Reads return zeros, writes have no effect.
0	G1_SAMP_DIS_EN		Group1 sample cap discharge enable. User or privileged mode read/write:
		0	Group1 sample cap discharge mode is disabled.
		1	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.

15.10.47 ADC Group2 Sample Cap Discharge Control Register (ADG2SAMPDISEN)

Figure 15-64 and Table 15-49 describe the ADG2SAMPDISEN register.

Figure 15-64. ADC Group2 Sample Cap Discharge Control Register (ADG2SAMPDISEN) [offset = 0x124]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
G2_SAMP_DIS_CYC								Reserved							G2_SAMP_DIS_EN
RW-0								R-0							RW-0

R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

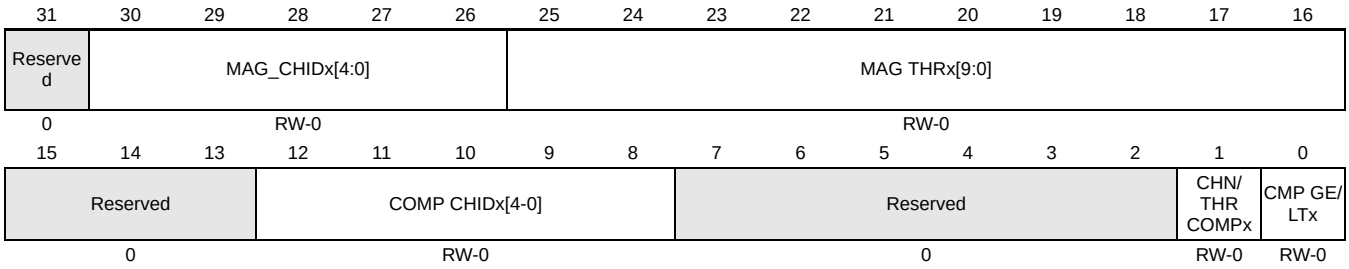
Table 15-49. ADC Group2 Sample Cap Discharge Control Register (ADG2SAMPDISEN) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved	0	Reads return zeros, writes have no effect.
15–8	G2_SAMP_DIS_CYC		Group2 sample cap discharge cycles. These bits specify the ADC internal sampling capacitor discharge duration before sampling the input channel voltage: (G2_SAMP_DIS_CYC) * $t_{C(ADCLK)}$. where $t_{C(ADCLK)}$ is the ADCLK clock period in ns.
7–1	Reserved	0	Reads return zeros, writes have no effect.
0	G2_SAMP_DIS_EN		Group2 sample cap discharge enable. User or privileged mode read/write:
		0	Group2 sample cap discharge mode is disabled.
		1	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.

15.10.48 ADC Magnitude Compare Interrupt Control Registers (ADMAGINTxCR)

Figure 15-65 and Table 15-50 describe the ADMAGINTxCR registers. This device supports three magnitude compare interrupts.

Figure 15-65. ADC Magnitude Compare Interruptx Control Registers (ADMAGINTxCR) [offset = 0x128, 0x130, 0x138, 0x140, 0x148, and 0x150]



R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-50. ADC Magnitude Compare Interruptx Control Register (ADMAGINTxCR) Field Descriptions

Bit	Name	Value	Description
31	Reserved	0	Reads return zeros, writes have no effect.
30–26	MAG_CHIDx[4:0]		These bits specify the channel number from 0 to 15 for which the conversion result needs to be monitored by the ADC.
25–16	MAG_THRx[9:0]		These bits specify the 10-bit compare value which the ADC will use for the comparison with the MAG_CHIDx channel's conversion result.
15–13	Reserved	0	Reads return zeros, writes have no effect.
12–8	COMP_CHIDx[4:0]		These bits specify the channel number from 0 to 15 whose last conversion result is compared with the MAG_CHIDx channel's conversion result.
7–2	Reserved	0	Reads return zeros, writes have no effect.
1	CHN/ THR_COMPx		Channel OR Threshold comparison. User or privileged mode read/write: 0 The ADC module will compare the MAG_CHIDx channel's conversion result with the fixed threshold value specified by the MAG_THRx field. 1 The ADC module will compare the MAG_CHIDx channel's conversion result with the last conversion result for the COMP_CHIDx channel. Both the MAG_CHIDx and the COMP_CHIDx channel must have been converted at least once for the ADC to perform the comparison.

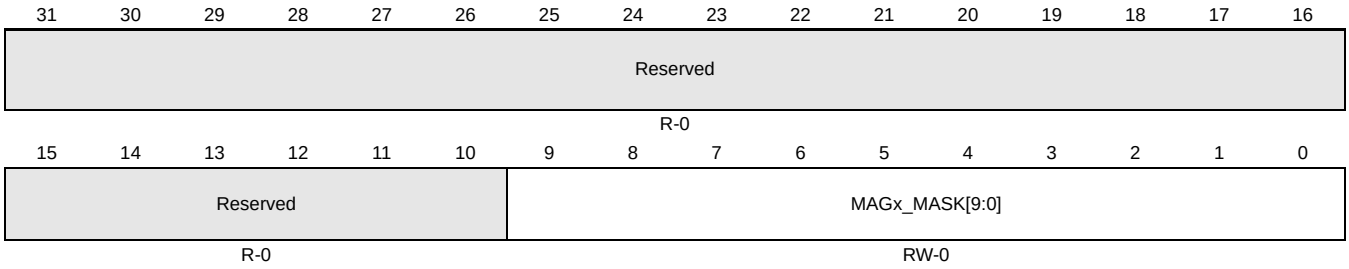
Table 15-50. ADC Magnitude Compare Interruptx Control Register (ADMAGINTxCR) Field Descriptions

Bit	Name	Value	Description
0	CMP_GE/LTx		“Greater than or equal to” OR “Less than” comparison operator. User or privileged mode read/write:
		0	The ADC module will check if the conversion result is lower than the reference value (fixed threshold or COMP_CHIDx conversion result).
		1	The ADC module will check if the conversion result is greater than or equal to the reference value (fixed threshold or COMP_CHIDx conversion result).

15.10.49 ADC Magnitude Compare Mask (ADMAGxMASK)

Figure 15-66 and Table 15-51 describe the ADMAGxMASK registers. There are three such registers for the three magnitude compare interrupts.

Figure 15-66. ADC Magnitude Compare Interrupt Mask (ADMAGxMASK) [offset = 0x12C, 0x134, 0x13C, 0x144, 0x14C and 0x154]



R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

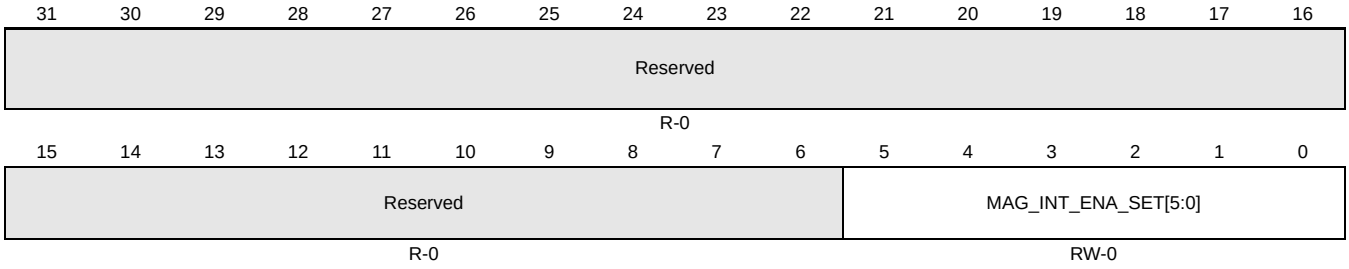
Table 15-51. ADC Magnitude Compare Interrupt Mask (ADMAGINTxMASK) Field Descriptions

Bit	Name	Value	Description
31–10	Reserved	0	Reads return zeros, writes have no effect.
9–0	MAGx_MASK[9:0]		These bits specify the mask for the comparison in order to generate the magnitude compare interrupt # x. User or privileged mode read/write:
		0	The ADC module will not mask the corresponding bit for the comparison.
		1	The ADC module will mask the corresponding bit for the comparison.

15.10.50 ADC Magnitude Compare Interrupt Enable Set (ADMAGINTENASET)

Figure 15-67 and Table 15-52 describe the ADMAGINTENASET register.

Figure 15-67. ADC Magnitude Compare Interrupt Enable Set (ADMAGINTENASET) [offset = 0x158]



R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

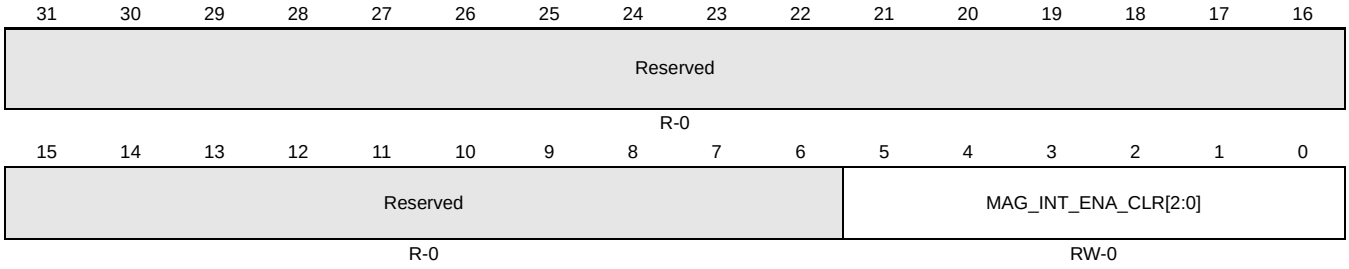
Table 15-52. ADC Magnitude Compare Interrupt Enable Set (ADMAGINTENASET) Field Descriptions

Bit	Name	Value	Description
31–3	Reserved	0	Reads return zeros, writes have no effect.
5–0	MAG_INT_ENA_SET[5:0]		Each of these three bits, when set, enable the corresponding magnitude compare interrupt. Bit0 controls magnitude compare interrupt 1. Bit1 controls magnitude compare interrupt 2. Bit2 controls magnitude compare interrupt 3. Bit3 controls magnitude compare interrupt 4. Bit4 controls magnitude compare interrupt 5. Bit5 controls magnitude compare interrupt 6. Any operation mode read/write for each bit:
		0	The enable status of the corresponding magnitude compare interrupt is left unchanged.
		1	The corresponding magnitude compare interrupt is enabled.

15.10.51 ADC Magnitude Compare Interrupt Enable Clear (ADMAGINTENACLR)

Figure 15-68 and Table 15-53 describe the ADMAGINTENACLR register.

Figure 15-68. ADC Magnitude Compare Interrupt Enable Clear (ADMAGINTENACLR) [offset = 0x15C]



R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

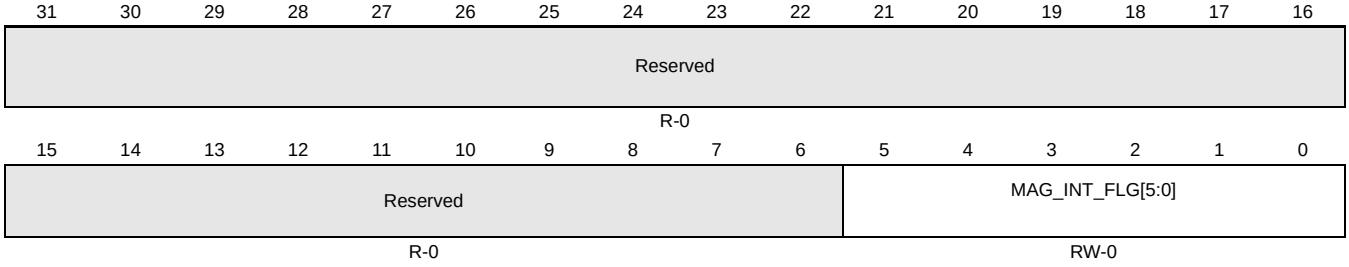
Table 15-53. ADC Magnitude Compare Interrupt Enable Clear (ADMAGINTENACLR) Field Descriptions

Bit	Name	Value	Description
31–3	Reserved	0	Reads return zeros, writes have no effect.
5–0	MAG_INT_ENA_ CLR[5:0]		Each of these three bits, when set, disable the corresponding magnitude compare interrupt. Bit0 controls magnitude compare interrupt 1. Bit1 controls magnitude compare interrupt 2. Bit2 controls magnitude compare interrupt 3. Bit3 controls magnitude compare interrupt 4. Bit4 controls magnitude compare interrupt 5. Bit5 controls magnitude compare interrupt 6. Any operation mode read/write for each bit:
		0	The enable status of the corresponding magnitude compare interrupt is left unchanged.
		1	The corresponding magnitude compare interrupt is disabled.

15.10.52 ADC Magnitude Compare Interrupt Flag (ADMAGINTFLG)

Figure 15-69 and Table 15-54 describe the ADMAGINTFLG register.

Figure 15-69. ADC Magnitude Compare Interrupt Flag (ADMAGINTFLG) [offset = 0x160]



R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

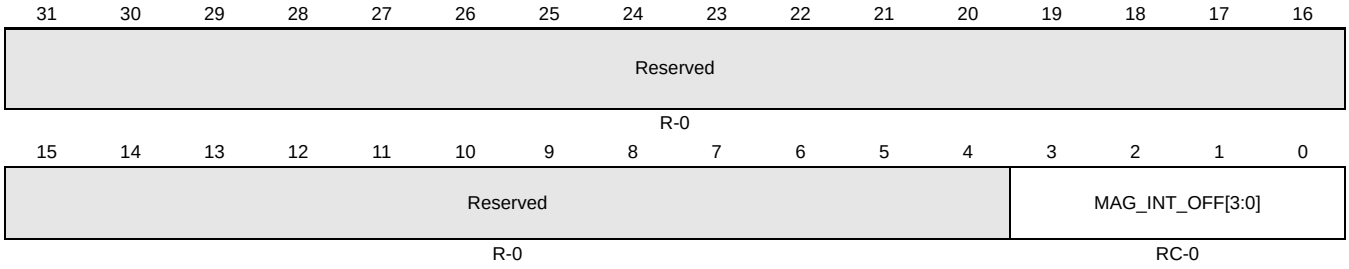
Table 15-54. ADC Magnitude Compare Interrupt Flag (ADMAGINTFLG) Field Descriptions

Bit	Name	Value	Description
31–3	Reserved	0	Reads return zeros, writes have no effect.
5–0	MAG_INT_FLG [5:0]		Magnitude Compare Interrupt Flags. These bits can be 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 will be generated if enabled. Bit0 is the flag of magnitude compare interrupt 1. Bit1 is the flag of magnitude compare interrupt 2. Bit2 is the flag of magnitude compare interrupt 3. Bit3 is the flag of magnitude compare interrupt 4. Bit4 is the flag of magnitude compare interrupt 5. Bit5 is the flag of magnitude compare interrupt 6. Any operation mode, for each bit:
		0	Read: The condition for the corresponding magnitude threshold interrupt has NOT been met. Write: The corresponding flag is left unchanged.
		1	Read: The condition for the corresponding magnitude threshold interrupt has been met. Write: The corresponding flag is cleared. The flag can also be cleared by reading from the magnitude compare interrupt offset register.

15.10.53 ADC Magnitude Compare Interrupt Offset (ADMAGINTOFF)

Figure 15-70 and Table 15-55 describe the ADMAGINTOFF register.

Figure 15-70. ADC Magnitude Compare Interrupt Offset (ADMAGINTOFF) [offset = 0x164]



R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

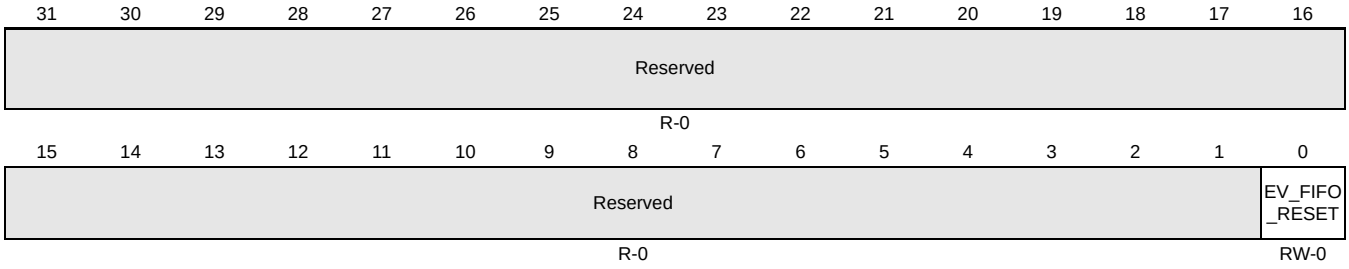
Table 15-55. ADC Magnitude Compare Interrupt Offset (ADMAGINTOFF) Field Descriptions

Bit	Name	Value	Description
31–4	Reserved	0	Reads return zeros, writes have no effect.
3–0	MAG_INT_OFF [3:0]		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. User or privileged mode read:
		0000b	No magnitude compare interrupt is pending.
		0001b	Magnitude compare interrupt # 1 is pending.
		0010b	Magnitude compare interrupt # 2 is pending.
		0011b	Magnitude compare interrupt # 3 is pending.
		0100b	Magnitude compare interrupt # 4 is pending.
		0101b	Magnitude compare interrupt # 5 is pending.
		0110b	Magnitude compare interrupt # 6 is pending.
		0111b	Invalid

15.10.54 ADC Event Group FIFO Reset Control Register (ADEVFIFORESETCR)

Figure 15-71 and Table 15-56 describe the ADEVFIFORESETCR register.

Figure 15-71. ADC Event Group FIFO Reset Control Register (ADEVFIFORESETCR) [offset = 0x168]



R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

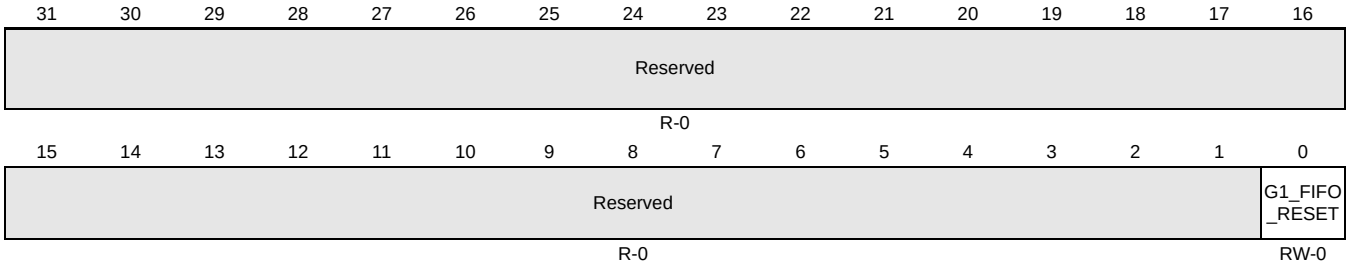
Table 15-56. ADC Event Group FIFO Reset Control Register (ADEVFIFORESETCR) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved	0	Reads return zeros, writes have no effect.
0	EV_FIFO_RESET		ADC Event Group FIFO Reset.The application can set 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. When this bit is set to ‘1’, 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 will always be read as a ‘0’. The EV_FIFO_RESET bit will only have 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 can be 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’.

15.10.55 ADC Group1 FIFO Reset Control Register (ADG1FIFORESETCR)

Figure 15-72 and Table 15-57 describe the ADG1FIFORESETCR register.

Figure 15-72. ADC Group1 FIFO Reset Control Register (ADG1FIFORESETCR) [offset = 0x16C]



R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

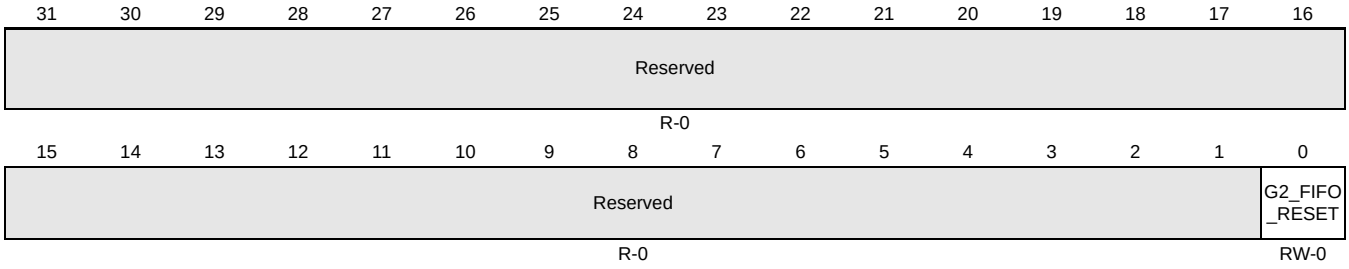
Table 15-57. ADC Group1 FIFO Reset Control Register (ADG1FIFORESETCR) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved	0	Reads return zeros, writes have no effect.
0	G1_FIFO_RESET		ADC Group1 FIFO Reset.The application can set this bit in case of an over-run condition. This allows the ADC module to overwrite the contents of the Group1 results memory starting from the first location. When this bit is set to ‘1’, 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 will always be read as a ‘0’. The G1_FIFO_RESET bit will only have 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 can be 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’.

15.10.56 ADC Group2 FIFO Reset Control Register (ADG2FIFORESETCR)

Figure 15-73 and Table 15-58 describe the ADG2FIFORESETCR register.

Figure 15-73. ADC Group2 FIFO Reset Control Register (ADG2FIFORESETCR) [offset = 0x170]



R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

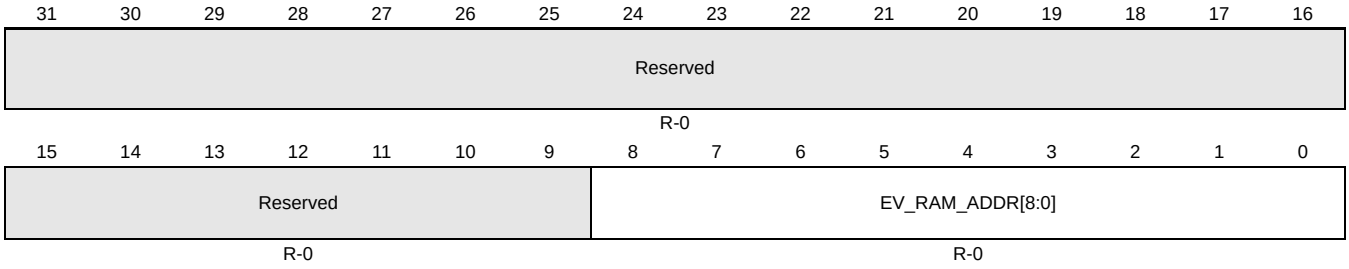
Table 15-58. ADC Group2 FIFO Reset Control Register (ADG2FIFORESETCR) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved	0	Reads return zeros, writes have no effect.
0	G2_FIFO_RESET		ADC Group2 FIFO Reset.The application can set this bit in case of an over-run condition. This allows the ADC module to overwrite the contents of the Group2 results memory starting from the first location. When this bit is set to ‘1’, 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 will always be read as a ‘0’. The G2_FIFO_RESET bit will only have 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 can be 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’.

15.10.57 ADC Event Group RAM Write Address (ADEVRAMWRADDR)

Figure 15-74 and Table 15-59 describe the ADEVRAMWRADDR register.

Figure 15-74. ADC Event Group RAM Write Address (ADEVRAMWRADDR) [offset = 0x174]



R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

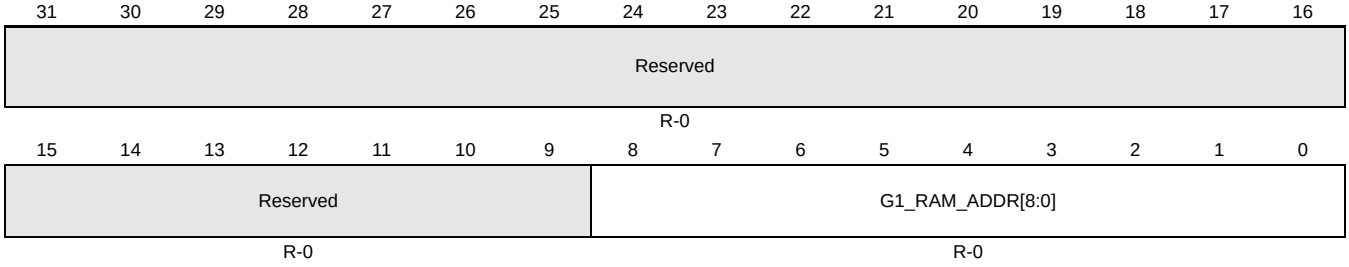
Table 15-59. ADC Event Group RAM Write Address (ADEVRAMWRADDR) Field Descriptions

Bit	Name	Value	Description
31–9	Reserved	0	Reads return zeros, writes have no effect.
8–0	EV_RAM_ADDR [8:0]		Event Group results memory write pointer. This field shows the address of the location where the next Event Group conversion result will be stored. This is specified in terms of the buffer number. The application can read this register to determine the number of valid Event Group conversion results available till that time.

15.10.58 ADC Group1 RAM Write Address (ADG1RAMWRADDR)

Figure 15-75 and Table 15-60 describe the ADG1RAMWRADDR register.

Figure 15-75. ADC Group1 RAM Write Address (ADG1RAMWRADDR) [offset = 0x178]



R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

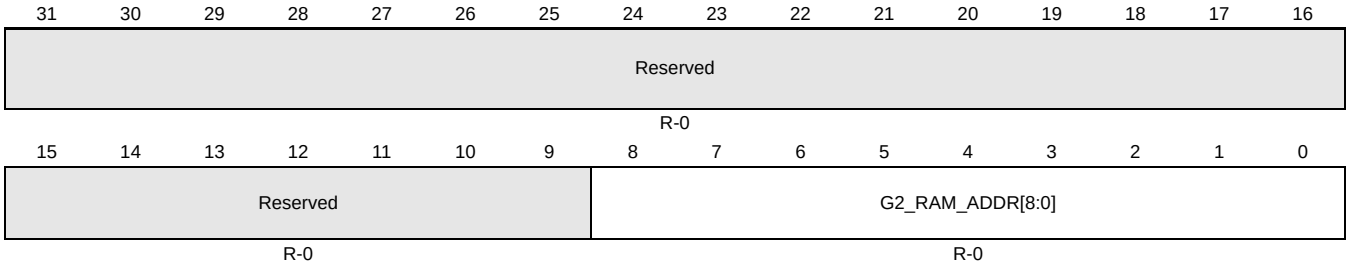
Table 15-60. ADC Group1 RAM Write Address (ADG1RAMWRADDR) Field Descriptions

Bit	Name	Value	Description
31–9	Reserved	0	Reads return zeros, writes have no effect.
8–0	G1_RAM_ADDR [8:0]		Group1 results memory write pointer. This field shows the address of the location where the next Group1 conversion result will be stored. This is specified in terms of the buffer number. The application can read this register to determine the number of valid Group1 conversion results available till that time.

15.10.59 ADC Group2 RAM Write Address (ADG2RAMWRADDR)

Figure 15-76 and Table 15-61 describe the ADG2RAMWRADDR register.

Figure 15-76. ADC Group2 RAM Write Address (ADG2RAMWRADDR) [offset = 0x17C]



R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

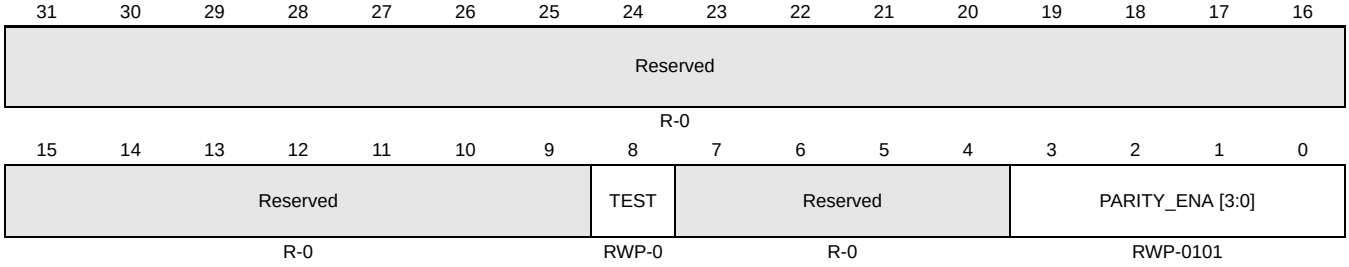
Table 15-61. ADC Group2 RAM Write Address (ADG2RAMWRADDR) Field Descriptions

Bit	Name	Value	Description
31–9	Reserved	0	Reads return zeros, writes have no effect.
8–0	G2_RAM_ADDR [8:0]		Group2 results memory write pointer. This field shows the address of the location where the next Group2 conversion result will be stored. This is specified in terms of the buffer number. The application can read this register to determine the number of valid Group2 conversion results available till that time.

15.10.60 ADC Parity Control Register (ADPARCR)

Figure 15-77 and Table 15-62 describe the ADPARCR register.

Figure 15-77. ADC Parity Control Register (ADPARCR) [offset = 0x180]



R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

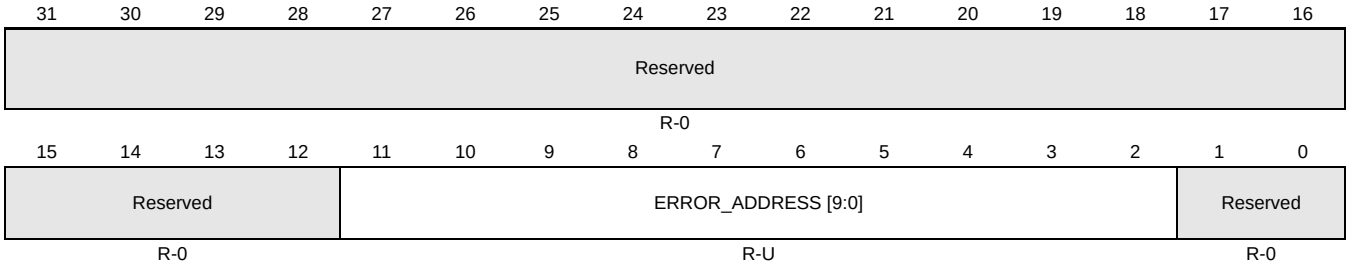
Table 15-62. ADC Parity Control Register (ADPARCR) Field Descriptions

Bit	Name	Value	Description
31–9	Reserved	0	Reads return zeros, writes have no effect.
8	TEST		This bit maps the parity bits into the ADC results FIFO RAM frame so that the application can access them. Any operation mode read, privileged mode write:
		0	The parity bits are not memory-mapped.
		1	The parity bits are memory mapped.
7–4	Reserved	0	Reads return zeros, writes have no effect.
3–0	PARITY_ENA [3:0]		Enable/disable parity checking. These bits enable/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 ESM module. Any operation mode read, privileged mode write:
		0101b	Parity check is disabled.
		Any other	Parity check is enabled. **Note: It is recommended to write 1010b to enable parity, to guard against soft error from flipping this field to a disabled state.**

15.10.61 ADC Parity Error Address (ADPARADDR)

Figure 15-78 and Table 15-63 describe the ADPARCR register.

Figure 15-78. ADC Parity Error Address (ADPARADDR) [offset = 0x184]



R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-63. ADC Parity Error Address (ADPARADDR) Field Descriptions

Bit	Name	Value	Description
31–12	Reserved	0	Reads return zeros, writes have no effect.
11–2	ERROR_ADDRESS [9:0]		These bits hold the address of the first parity error generated in the ADC results FIFO RAM. This error address is frozen from being updated until it is read by the application. During debugger memory reads, this address is maintained frozen even when read.
1–0	Reserved	0	Reads return zeros, writes have no effect. Reading [11:0] provides the 32-bit aligned address.

Real-Time Interrupt (RTI) Module

This chapter describes the functionality of the real-time interrupt (RTI) module implemented on the TMS470M Series of microcontrollers. It is specifically designed to support time-triggered operating systems and real-time operating systems.

Topic **Page**

16.1 Introduction and Feature Overview 856
16.2 Module Operation 857
16.3 Control Registers 861

16.1 Introduction and Feature Overview

This section describes the purpose, features, register mapping, and industry standard compliance.

16.1.1 Purpose

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.

16.1.2 Main Features

The RTI module has the following features:

- Four configurable compare registers for generating interrupts. Each register can work with counter block 0 or counter block 1.
- Automatic update of compare register on compare match to generate periodic interrupts
- Fast enabling/disabling of interrupts
- Digital watchdog (DWD)
- RTI clock input selectable in the system module

16.1.3 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).

16.2 Module Operation

The following sections describe the operation of the RTI module.

16.2.1 Counter Operation

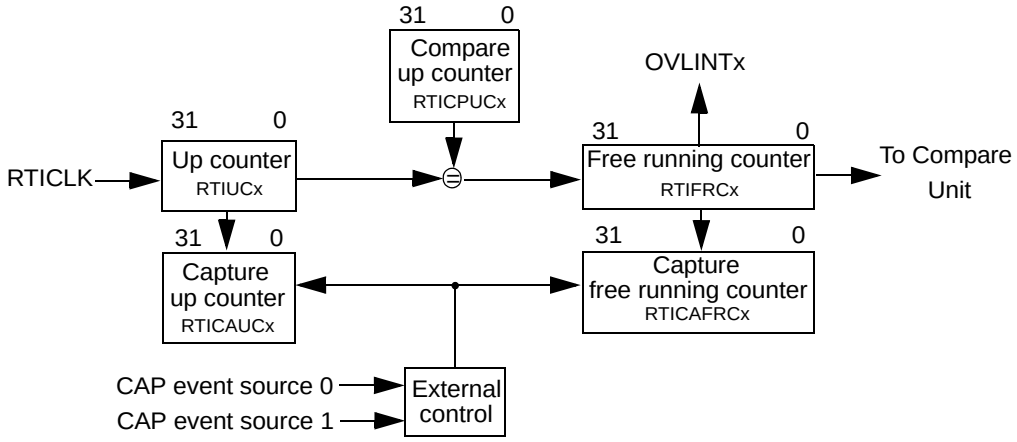
- One 32-bit prescale counter (RTIUCx)
- One 32-bit free running counter (RTIFRCx)

The RTIUCx is driven by the RTICLK and counts up until the compare value in the compare up counter register (RTICPUCx) is reached. When the compare matches, RTIFRCx is incremented and RTIUCx is reset to 0. If RTIFRCx overflows, an interrupt is generated to the vectored interrupt manager (M3VIM). This process means that the value set in the compare up counter (RTICPUCx) prescales the RTI clock. The resulting formula for the frequency of the free running counter (RTIFRCx) is:

$f_{RTIFRCx} = f_{RTICLK}/(RTICPUCx+1)$

Note:
There is one exception. If CPUCx = 0, then $f_{FRCx} = RTICLK/2^{32}$

Figure 16-1. Counter Block Diagram



16.2.1.1 Counter Read Consistency

To ensure the consistency of the counters, when both counter values must be determined, RTIFRCx must be read first. This priority will ensure that at the CPU read cycle of RTIFRCx, the up counter value is stored in the counter register. The second read is done on the up counter register (RTIUCx), which then holds the value of the counter cycle of the corresponding read on the free running counter register (RTIFRCx).

16.2.1.2 Capture Feature

Both blocks also provide a capture feature on external events. Two capture sources can trigger the capture event. Which source triggers the block is configurable. The sources originate from the M3VIM, to generate a capture event when one of the peripheral modules has generated an interrupt. The peripheral that can generate an event is configured in the vectored interrupt manager (M3VIM).

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 order of the actual counters. Therefore, RTICAFRCx must be read first and the RTICAUCx registers must be read after the RTICAFRCx value has been determined. While RTICAFRCx is read, the RTICAUCx value is loaded into a shadow register to ensure data consistency if another capture event occurs during the two reads of the captured data. If the application fails to read the two registers before a second capture event happens, the previous data will be overwritten.

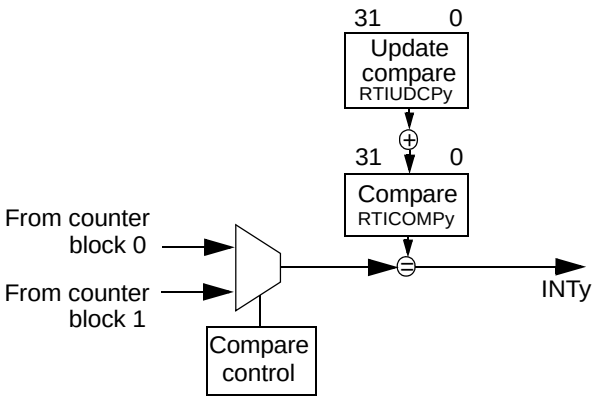
16.2.1.3 Interrupt Requests

To generate interrupt requests to the M3VIM, there are four compare registers (RTICOMPy). Each of the compare registers can be configured to work on either RTIFRC0 or RTIFRC1. When the counter value matches the compare value, an interrupt is generated. All compares provide an additional feature, with an automatic addition of the compare value with the value stored in the update compare (RTIUDCPy) register when the compare matches. This allows periodic interrupt requests to be generated without having to update the compare value by software. The period of the generated interrupt request is given by the formula:

$t_{COMPx} = t_{RTICLK}$ x (RTICPUCy+1) x RTIUDCPy

Another interrupt that can be generated is the overflow interrupt (OVLINTx) in case the RTIFRCx counter overflows.

Figure 16-2. Compare Block Diagram



The interrupts can be enabled in the RTISETINT register and disabled in the RTICLEARINT register. The RTIFLAG register shows the pending interrupts.

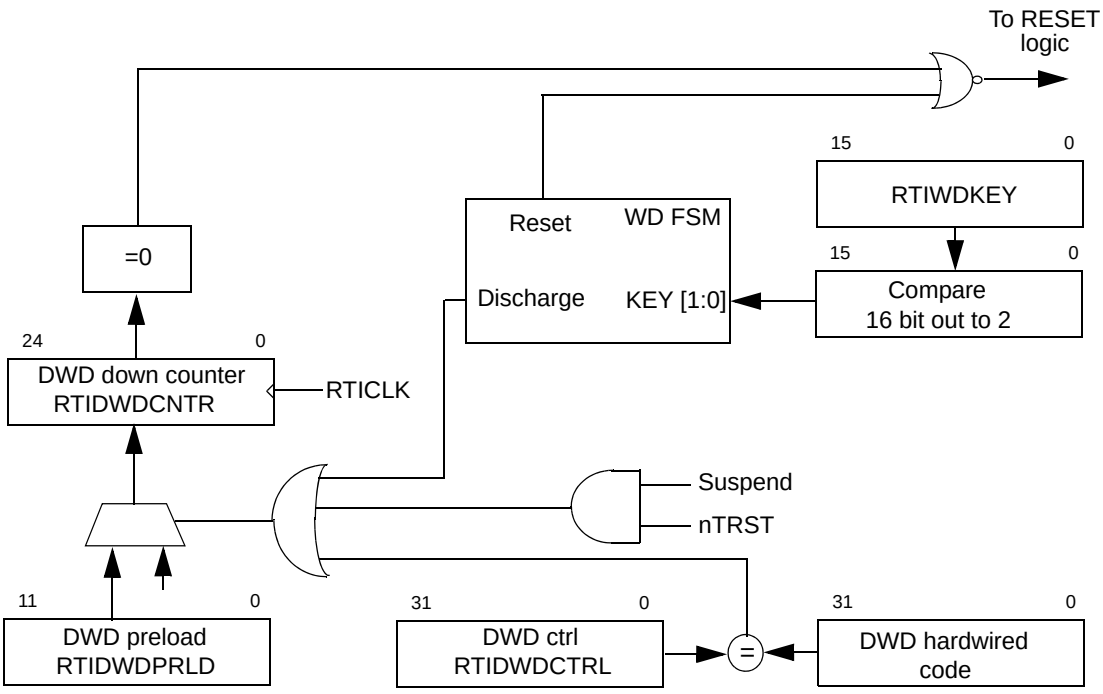
16.2.2 Clock Domain

For RTICLK definition, please see Section 1.5.2, "Clock Domains" in the system module.

16.2.3 Digital Watchdog (DWD)

The digital watchdog (DWD) generate resets to prevent runaway code. Figure 16-3 illustrates the DWD.

Figure 16-3. Digital Watchdog



Note:
It has to be taken into account that the write to the RTIWDKEY register takes 2 VCLK cycles.

16.2.3.1 DWD

The DWD also generates resets after a programmable period, or if no correct key sequence was written to the RTIWDKEY register.

The DWD is disabled by default. If it should be used, it must be enabled by writing a 32-bit value, which is the inverted value of the hardwired code in the module, to the RTIDWDCTRL register.

Note:
Once the DWD is enabled, it cannot be disabled.

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 will occur immediately. A reset will also be generated when the DWD down counter is decremented to 0.

While the device is in suspend mode (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.

The expiration time of the DWD down counter can be determined with the following equation:

$texp = (RTIDWDPRLD + 1) \times 2^{13}/RTICLK$

where

RTIDWDPRLD = 0...4095

Note:
It has to be taken into account that the write to the RTIWDKEY register takes 3 VCLK cycles.

16.3 Control Registers

Figure 16-4 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 0xFFFF FC00. The address locations not listed, are reserved..

Figure 16-4. RTI 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
0x00 RTIGCTRL Page 865	Reserved															
	COS	Reserved													CNT1 EN	CNT0 EN
0x08 RTICAPCTRL Page 866	Reserved															
	Reserved														CAP CNTR1	CAP CNTR0
0x0C RTICOMPCTRL Page 867	Reserved															
	Reserved			COMP SEL3	Reserved			COMPS EL2	Reserved			COMPS EL1	Reserved			COMPS EL0
0x10 RTIFRC0 Page 868	FRC0(31–16)															
	FRC0(15–0)															
0x14 RTIUC0 Page 869	UC0(31–16)															
	UC0(15–0)															
0x18 RTICPUC0 Page 870	CPUC0(31–16)															
	CPUC0(15–0)															
0x20 RTICAFRC0 Page 871	CAFRC0(31–16)															
	CAFRC0(15–0)															
0x24 RTICAUC0 Page 872	CAUC0(31–16)															
	CAUC0(15–0)															

Figure 16-4. RTI Registers (Continued)

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
0x30 RTIFRC1 Page 873	FRC1(31–16)															
	FRC1(15–0)															
0x34 RTIUC1 Page 874	UC1(31–16)															
	UC1(15–0)															
0x38 RTICPUC1 Page 875	CPUC1(31–16)															
	CPUC1(15–0)															
0x40 RTICAFRC1 Page 876	CAFRC1(31–16)															
	CAFRC1(15–0)															
0x44 RTICAUC1 Page 877	CAUC1(31–16)															
	CAUC1(15–0)															
0x50 RTICOMP0 Page 878	COMP0(31–16)															
	COMP0(15–0)															
0x54 RTIUDCP0 Page 879	UDCP0(31–16)															
	UDCP0(15–0)															
0x58 RTICOMP1 Page 880	COMP1(31–16)															
	COMP1(15–0)															
0x5C RTIUDCP1 Page 881	UDCP1(31–16)															
	UDCP1(15–0)															

Figure 16-4. RTI Registers (Continued)

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
0x60 RTICOMP2 Page 882	COMP2(31–16)															
	COMP2(15–0)															
0x64 RTIUDCP2 Page 883	UDCP2(31–16)															
	UDCP2(15–0)															
0x68 RTICOMP3 Page 884	COMP3(31–16)															
	COMP3(15–0)															
0x6C RTIUDCP3 Page 885	UDCP3(31–16)															
	UDCP3(15–0)															
0x80 RTISETINT Page 886	Reserved													SET OVL1 INT	SET OVL0 INT	Reserved
	Reserved												SET INT3	SET INT2	SET INT1	SET INT0
0x84 RTICLEARINT Page 888	Reserved													CLEAR OVL1 INT	CLEAR OVL0 INT	Reserved
	Reserved												CLEAR INT3	CLEAR INT2	CLEAR INT1	CLEAR INT0
0x88 RTIINTFLAG Page 890	Reserved													OVL1 INT	OVL0 INT	Reserved
	Reserved												INT3	INT2	INT1	INT0
0x90 RTIDWDCTRL Page 892	DWDCTRL[31:16]															
	DWDCTRL[15:0]															
0x94 RTIDWDPRLD Page 893	Reserved															
	Reserved					DWDPRLD(11–0)										

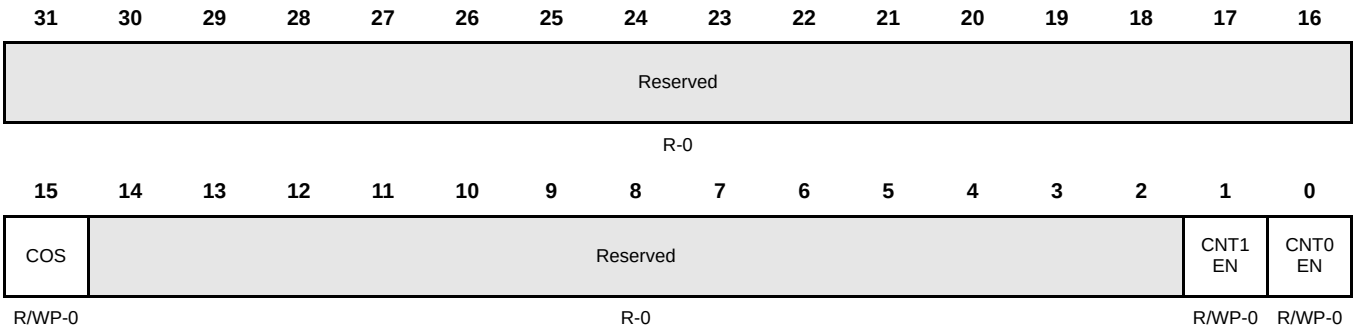
Figure 16-4. RTI Registers (Continued)

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
0x98 RTIWDSTATUS Page 894	Reserved															
0x9C RTIWDKEY Page 895	Reserved															
	WDKEY(15–0)															
0xA0 RTIDWDCNTR Page 896	Reserved							DWDCNTR(24–16)								
	DWDCNTR(15–0)															

16.3.1 RTI Global Control Register (RTIGCTRL)

The global control register starts/stops the counters. This register is shown in Figure 16-5 and described in Table 16-1.

Figure 16-5. RTI Global Control Register (RTIGCTRL) [offset = 0x00]



R = Read, WP = Write in privileged mode only, -*n* = Value after reset

Table 16-1. RTI Global Control Register (RTIGCTRL) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Reads return 0 and writes have no effect.
15	COS		Continue on suspend. This bit determines if both counters are stopped when the device goes into suspend mode or if they continue counting.
		0	Counters are stopped while in suspend mode.
		1	Counters are running while in suspend mode.
14–2	Reserved		Reads return 0 and 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 debug mode, the synchronization may be lost depending on the COS setting in the RTI module and the debug mode behavior of the communications controller.

16.3.2 RTI Capture Control Register (RTICAPCTRL)

The capture control register controls the capture source for the counters. This register is shown in Figure 16-6 and described in Table 16-2.

Figure 16-6. RTI Capture Control Register (RTICAPCTRL) [offset = 0x08]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														CAP CNTR1	CAP CNTR0
R-0														R/WP-0	R/WP-0

R = Read, WP = Write in privileged mode only, -*n* = Value after reset

Table 16-2. RTI Capture Control Register (RTICAPCTRL) Field Descriptions

Bit	Name	Value	Description
31–2	Reserved		Reads return 0 and 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.

16.3.3 RTI Compare Control Register (RTICOMPCTRL)

The compare control register controls the source for the compare registers. This register is shown in Figure 16-7 and described in Table 16-7.

Figure 16-7. RTI Compare Control Register (RTICOMPCTRL) [offset = 0x0C]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			COMPSEL3	Reserved			COMPSEL2	Reserved			COMPSEL1	Reserved			COMPSEL0
R-0			R/WP-0	R-0			R/WP-0	R-0			R/WP-0	R-0			R/WP-0

R = Read, WP = Write in privilege mode only, -*n* = Value after reset

Table 16-3. RTI Compare Control Register (RTICOMPCTRL) Field Descriptions

Bit	Name	Value	Description
31–13	Reserved		Reads return 0 and 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		Reads return 0 and 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		Reads return 0 and 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		Reads return 0 and 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.

16.3.4 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 16-8 and described in Table 16-4.

Figure 16-8. RTI Free Running Counter 0 Register (RTIFRC0) [offset = 0x10]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FRC0(31–16)															
R/WP-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FRC0(15–0)															
R/WP-0															

R = Read, WP = Write in privileged mode only, -*n* = Value after reset

Table 16-4. RTI Free Running Counter 0 Register (RTIFRC0) Field Descriptions

Bit	Name	Value	Description
31–0	FRC0(31–0)	0–FFFF FFFFh	Free running counter 0. This registers holds the current value of the free running counter 0 (RTIFRC0) and will be updated continuously. 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.**

16.3.5 RTI Up Counter 0 Register (RTIUC0)

The up counter 0 register holds the current value of prescale counter. This register is shown in Figure 16-9 and described in Table 16-5.

Figure 16-9. RTI Up Counter 0 Register (RTIUC0) [offset = 0x14]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UC0(31–16)															
R/WP-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
UC0(15–0)															
R/WP-0															

R = Read, WP = Write in privileged mode only, -*n* = Value after reset

Table 16-5. RTI Up Counter 0 Register (RTIUC0) Field Descriptions

Bit	Name	Value	Description
31–0	UC0(31–0)	0–FFFF FFFFh	Up counter 0. This register holds the current value of the up counter 0 (RTIUC0) 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.**

16.3.6 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 16-10 and described in Table 16-6.

Figure 16-10. RTI Compare Up Counter 0 Register (RTICPUC0) [offset = 0x18]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CPUC0(31–16)															
R/WP-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CPUC0(15–0)															
R/WP-0															

R = Read, WP = Write in privileged mode only, -*n* = Value after reset

Table 16-6. RTI Compare Up Counter 0 Register (RTICPUC0) Field Descriptions

Bit	Name	Value	Description

31–0	CPUC0(31–0)	0–FFFF FFFFh	Compare up counter 0. This register holds the value that is compared with the up counter 0 (RTIUC0). 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
			$f_{FRC0} = RTICLK/2^{32}$
			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.

16.3.7 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 16-11 and described in Table 16-7.

Figure 16-11. RTI Capture Free Running Counter 0 Register (RTICAFRC0) [offset = 0x20]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CAFRC0(31–16)															
R-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CAFRC0(15–0)															
R-0															

R = Read, WP = Write in privileged mode only, -*n* = Value after reset

Table 16-7. RTI Capture Free Running Counter 0 Register (RTICAFRC0) Field Descriptions

Bit	Name	Value	Description
31–0	CAFRC0(31–0)	0–FFFF FFFFh	Capture free running counter 0. This register captures the current value of the free running counter 0 (RTIFRC0) when a event occurs, controlled by the external capture control block. A read of this register returns the value of RTIFRC0 on a capture event.

16.3.8 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 16-12 and described in Table 16-8.

Figure 16-12. RTI Capture Up Counter 0 Register (RTICAUC0) [offset = 0x24]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CAUC0(31–16)															
R-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CAUC0(15–0)															
R-0															

R = Read, WP = Write in privileged mode only, -*n* = Value after reset

Table 16-8. RTI Capture Up Counter 0 Register (RTICAUC0) Field Descriptions

Bit	Name	Value	Description
31–0	CAUC0(31–0)	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.

16.3.9 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 16-13 and described in Table 16-9.

Figure 16-13. RTI Free Running Counter 1 Register (RTIFRC1) [offset = 0x30]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FRC1(31–16)															
R/WP-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FRC1(15–0)															
R/WP-0															

R = Read, WP = Write in privilege mode only, -*n* = Value after reset

Table 16-9. RTI Free Running Counter 1 Register (RTIFRC1) Field Descriptions

Bit	Name	Value	Description
31–0	FRC1(31–0)	0–FFFF FFFFh	Free running counter 1. This register holds the current value of the free running counter 1 (RTIFRC1) 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.**

16.3.10 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 16-14 and described in Table 16-10.

Figure 16-14. RTI Up Counter 1 Register (RTIUC1) [offset = 0x34]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UC1(31–16)															
R/WP-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
UC1(15–0)															
R/WP-0															

R = Read, WP = Write in privilege mode only, -*n* = Value after reset

Table 16-10. RTI Up Counter 1 Register (RTIUC1) Field Descriptions

Bit	Name	Value	Description
31–0	UC1(31–0)	0–FFFF FFFFh	Up counter 1. This register holds the current value of the up counter 1 (RTIUC1 and RTIFRC1) 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.**

16.3.11 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 16-15 and described in Table 16-11.

Figure 16-15. RTI Compare Up Counter 1 Register (RTICPUC1) [offset = 0x38]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CPUC1(31–16)															
R/WP-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CPUC1(15–0)															
R/WP-0															

R = Read, WP = Write in privilege mode only, -*n* = Value after reset

Table 16-11. RTI Compare Up Counter 1 Register (RTICPUC1) Field Descriptions

Bit	Name	Value	Description

31–0	CPUC1(31–0)	0–FFFF FFFFh	Compare up counter 1. This register holds the compare value, which is compared with the up counter 1 (RTIUC1). When the compare matches, the free running counter 1 (RTIFRC1) is incremented. The up counter is set to zero 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}
			
			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.

16.3.12 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 16-16 and described in Table 16-12.

Figure 16-16. RTI Capture Free Running Counter 1 Register (RTICAFRC1) [offset = 0x40]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CAFRC1(31–16)															
R-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CAFRC1(15–0)															
R-0															

R = Read, -*n* = Value after reset

Table 16-12. RTI Capture Free Running Counter 1 Register (RTICAFRC1) Field Descriptions

Bit	Name	Value	Description
31–0	CAFRC1(31–0)	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.

16.3.13 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 16-17 and described in Table 16-13.

Figure 16-17. RTI Capture Up Counter 1 Register (RTICAUC1) [offset = 0x44]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CAUC1(31–16)															
R-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CAUC1(15–0)															
R-0															

R = Read, WP = Write in privilege mode only, -*n* = Value after reset

Table 16-13. RTI Capture Up Counter 1 Register (RTICAUC1) Field Descriptions

Bit	Name	Value	Description
31–0	CAUC1(31–0)	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.

16.3.14 RTI Compare 0 Register (RTICOMP0)

The compare 0 register holds the value to be compared with the counters. This register is shown in Figure 16-18 and described in Table 16-14.

Figure 16-18. RTI Compare 0 Register (RTICOMP0) [offset = 0x50]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMP0(31–16)															
R/WP-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
COMP0(15–0)															
R/WP-0															

R = Read, WP = Write in privileged mode only, -*n* = Value after reset

Table 16-14. RTI Compare 0 Register (RTICOMP0) Field Descriptions

Bit	Name	Value	Description
31–0	COMP0(31–0)	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. 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.

16.3.15 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 16-19 and described in Table 16-15.

Figure 16-19. RTI Update Compare 0 Register (RTIUDCP0) [offset = 0x54]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UDCP0(31–16)															
R/WP-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
UDCP0(15–0)															
R/WP-0															

R = Read, WP = Write in privileged mode only, -*n* = Value after reset

Table 16-15. RTI Update Compare 0 Register (RTIUDCP0) Field Descriptions

Bit	Name	Value	Description
31–0	UDCP0(31–0)	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 compare 0 register on the next compare match. A write to this register will provide a new update value.

16.3.16 RTI Compare 1 Register (RTICOMP1)

The compare 1 register holds the value to be compared to the counters. This register is shown in Figure 16-20 and described in Table 16-16.

Figure 16-20. RTI Compare 1 Register (RTICOMP1) [offset = 0x58]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMP1(31–16)															
R/WP-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
COMP1(15–0)															
R/WP-0															

R = Read, WP = Write in privileged mode only, -*n* = Value after reset

Table 16-16. RTI Compare 1 Register (RTICOMP1) Field Descriptions

Bit	Name	Value	Description
31–0	COMP1(31–0)	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. 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.

16.3.17 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 16-21 and described in Table 16-17.

Figure 16-21. RTI Update Compare 1 Register (RTIUDCP1) [offset = 0x5C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UDCP1(31–16)															
R/WP-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
UDCP1(15–0)															
R/WP-0															

R = Read, WP = Write in privileged mode only, -*n* = Value after reset

Table 16-17. RTI Update Compare 1 Register (RTIUDCP1) Field Descriptions

Bit	Name	Value	Description
31–0	UDCP1(31–0)	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.

16.3.18 RTI Compare 2 Register (RTICOMP2)

The compare 2 register holds the value to be compared to the counters. This register is shown in Figure 16-22 and described in Table 16-18.

Figure 16-22. RTI Compare 2 Register (RTICOMP2) [offset = 0x60]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMP2(31–16)															
R/WP-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
COMP2(15–0)															
R/WP-0															

R = Read, WP = Write in privileged mode only, -*n* = Value after reset

Table 16-18. RTI Compare 2 Register (RTICOMP2) Field Descriptions

Bit	Name	Value	Description
31–0	COMP2(31–0)	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. 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.

16.3.19 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 16-23 and described in Table 16-19.

Figure 16-23. RTI Update Compare 2 Register (RTIUDCP2) [offset = 0x64]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UDCP2(31–16)															
R/WP-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
UDCP2(15–0)															
R/WP-0															

R = Read, WP = Write in privileged mode only, -*n* = Value after reset

Table 16-19. RTI Update Compare 2 Register (RTIUDCP2) Field Descriptions

Bit	Name	Value	Description
31–0	UDCP2(31–0)	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.

16.3.20 RTI Compare 3 Register (RTICOMP3)

The compare 3 register holds the value to be compared to the counters. This register is shown in Figure 16-24 and described in Table 16-20.

Figure 16-24. RTI Compare 3 Register (RTICOMP3) [offset = 0x68]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMP3(31–16)															
R/WP-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
COMP3(15–0)															
R/WP-0															

R = Read, WP = Write in privileged mode only, -*n* = Value after reset

Table 16-20. RTI Compare 3 Register (RTICOMP3) Field Descriptions

Bit	Name	Value	Description
31–0	COMP3(31–0)	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. A read of this register will return the current compare value. A write to this register will provide a new compare value.

16.3.21 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 16-25 and described in Table 16-21.

Figure 16-25. RTI Update Compare 3 Register (RTIUDCP3) [offset = 0x6C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UDCP3(31–16)															
R/WP-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
UDCP3(15–0)															
R/WP-0															

R = Read, WP = Write in privileged mode only, -*n* = Value after reset

Table 16-21. RTI Update Compare 3 Register (RTIUDCP3) Field Descriptions

Bit	Name	Value	Description
31–0	UDCP3(31–0)	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.

16.3.22 RTI Set/Status Interrupt Register (RTISETINT)

This register prevents the necessity of a read-modify-write operation if a particular interrupt should be enabled. This register is shown in Figure 16-26 and described in Table 16-22.

Figure 16-26. RTI Set/Status Interrupt Control Register (RTISETINT) [offset = 0x80]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													SETOVL1INT	SETOVL0INT	Reserved
R-0													R/WP-0	R/WP-0	R-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												SETINT3	SETINT2	SETINT1	SETINT0
R-0												R/WP-0	R/WP-0	R/WP-0	R/WP-0

R = Read, WP = Write in privileged mode only, -*n* = Value after reset

Table 16-22. RTI Set/Status Interrupt Control Register (RTISETINT) Field Descriptions

Bit	Name	Value	Description
31–19	Reserved		Reads return 0 and writes have no effect.
18	SETOVL1INT		Set free running counter 1 overflow interrupt.
		0	*Read:* The interrupt is disabled. *Write:* The corresponding bit is unchanged.
		1	*Read or write:* The interrupt is enabled.
17	SETOVL0INT		Set free running counter 0 overflow interrupt.
		0	*Read:* The interrupt is disabled. *Write:* The corresponding bit is unchanged.
		1	*Read or write:* The interrupt is enabled.
16–4	Reserved		Reads return 0 and writes have no effect.
3	SETINT3		Set compare interrupt 3.
		0	*Read:* The interrupt is disabled. *Write:* The corresponding bit is unchanged.
		1	*Read or write:* The interrupt is enabled.
2	SETINT2		Set compare interrupt 2.
		0	*Read:* The interrupt is disabled. *Write:* The corresponding bit is unchanged.
		1	*Read or write:* The interrupt is enabled.

Table 16-22. RTI Set/Status Interrupt Control Register (RTISETINT) Field Descriptions (Continued)

Bit	Name	Value	Description
1	SETINT1		Set compare interrupt 1.
		0	*Read:* The interrupt is disabled. *Write:* The corresponding bit is unchanged.
		1	*Read or write:* The interrupt is enabled.
0	SETINT0		Set compare interrupt 0.
		0	*Read:* The interrupt is disabled. *Write:* The corresponding bit is unchanged.
		1	*Read or write:* The interrupt is enabled.

16.3.23 RTI Clear/Status Interrupt Register (RTICLEARINT)

This register prevents the necessity of a read-modify-write operation if a particular interrupt should be disabled. This register is shown in Figure 16-27 and described in Table 16-23.

Figure 16-27. RTI Clear/Status Interrupt Control Register (RTICLEARINT) [offset = 0x84]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													CLEAR OVL1 INT	CLEAR OVL0 INT	Reserved
R-0													R/WP-0	R/WP-0	R-0
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved												CLEAR INT3	CLEAR INT2	CLEAR INT1	CLEAR INT0
R-0												R/WP-0	R/WP-0	R/WP-0	R/WP-0

R = Read, WP = Write in privileged mode only, -*n* = Value after reset

Table 16-23. RTI Clear/Status Interrupt Control Register (RTICLEARINT) Field Descriptions

Bit	Name	Value	Description
31–19	Reserved		Reads return 0 and writes have no effect.
18	CLEAROVL1 INT		Clear free running counter 1 overflow interrupt.
		0	*Read:* The interrupt is disabled. *Write:* The corresponding bit is unchanged.
		1	*Read:* The interrupt is enabled. *Write:* The interrupt is disabled.
17	CLEAROVL0 INT		Clear free running counter 0 overflow interrupt.
		0	*Read:* The interrupt is disabled. *Write:* The corresponding bit is unchanged.
		1	*Read:* The interrupt is enabled. *Write:* The interrupt is disabled.
16–4	Reserved		Reads return 0 and writes have no effect.
3	CLEARINT3		Set compare interrupt 3.
		0	*Read:* The interrupt is disabled. *Write:* The corresponding bit is unchanged.
		1	*Read:* The interrupt is enabled. *Write:* The interrupt is disabled.
2	CLEARINT2		Set compare interrupt 2.
		0	*Read:* The interrupt is disabled. *Write:* The corresponding bit is unchanged.
		1	*Read:* The interrupt is enabled. *Write:* The interrupt is disabled.

Table 16-23. RTI Clear/Status Interrupt Control Register (RTICLEARINT) Field Descriptions (Continued)

Bit	Name	Value	Description
1	CLEARINT1		Set compare interrupt 1.
		0	*Read:* The interrupt is disabled. *Write:* The corresponding bit is unchanged.
		1	*Read:* The interrupt is enabled. *Write:* The interrupt is disabled.
0	CLEARINT0		Set compare interrupt 0.
		0	*Read:* The interrupt is disabled. *Write:* The corresponding bit is unchanged.
		1	*Read:* The interrupt is enabled. *Write:* The interrupt is disabled.

16.3.24 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 16-28 and described in Table 16-24.

Figure 16-28. RTI Interrupt Flag Register (RTIINTFLAG) [offset = 0x88]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													OVL1 INT	OVL0 INT	Reserved
R-0													R/WP-0	R/WP-0	R-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												INT3	INT2	INT1	INT0
R-0												R/CP-0	R/CP-0	R/CP-0	R/CP-0

R = Read, CP = Clear in privileged mode only, U = Undefined; -*n* = Value after reset

Table 16-24. RTI Interrupt Flag Register (RTIINTFLAG) Field Descriptions

Bit	Name	Value	Description
31–19	Reserved		Reads return 0 and writes have no effect.
18	OVL1INT		Free running counter 1 overflow interrupt flag. This bit determines if an interrupt is pending.
		0	*Read:* No interrupt is pending. *Write:* The bit is unchanged.
		1	*Read:* An interrupt is pending. *Write:* The bit is set to 0.
17	OVL0INT		Free running counter 0 overflow interrupt flag. This bit determines if an interrupt is pending.
		0	*Read:* No interrupt is pending. *Write:* The bit is unchanged.
		1	*Read:* An interrupt is pending. *Write:* The bit is set to 0.
16–4	Reserved		Reads return 0 and writes have no effect.
3	INT3		Interrupt flag 3. These bits determine if an interrupt is pending.
		0	*Read:* No interrupt is pending. *Write:* The bit is unchanged.
		1	*Read:* An interrupt is pending. *Write:* The bit is set to 0.
2	INT2		Interrupt flag 2. These bits determine if an interrupt is pending.
		0	*Read:* No interrupt is pending. *Write:* The bit is unchanged.
		1	*Read:* An interrupt is pending. *Write:* The bit is set to 0.

Table 16-24. RTI Interrupt Flag Register (RTIINTFLAG) Field Descriptions (Continued)

Bit	Name	Value	Description
1	INT1		Interrupt flag 1. These bits determine if an interrupt is pending.
		0	*Read:* No interrupt is pending. *Write:* The bit is unchanged.
		1	*Read:* An interrupt is pending. *Write:* The bit is set to 0.
0	INT0		Interrupt flag 0. These bits determine if an interrupt is pending.
		0	*Read:* No interrupt is pending. *Write:* The bit is unchanged.
		1	*Read:* An interrupt is pending. *Write:* The bit is set to 0.

16.3.25 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. The RTIDWDCTRL register also implements a one-time-write constraint. That is, once the application writes to this register to enable the watchdog, all further writes are ignored.

This register is shown in Figure 16-28 and described in Table 16-24.

Figure 16-29. Digital Watchdog Control Register (RTIDWDCTRL) [offset = 0x90h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DWDCTRL[31:16]															
R/WP-0x5312															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DWDCTRL[15:0]															
R/WP-0xACED															

R = Read, W = Write, P = Privilege Mode, U = Undefined; -*n* = Value after reset

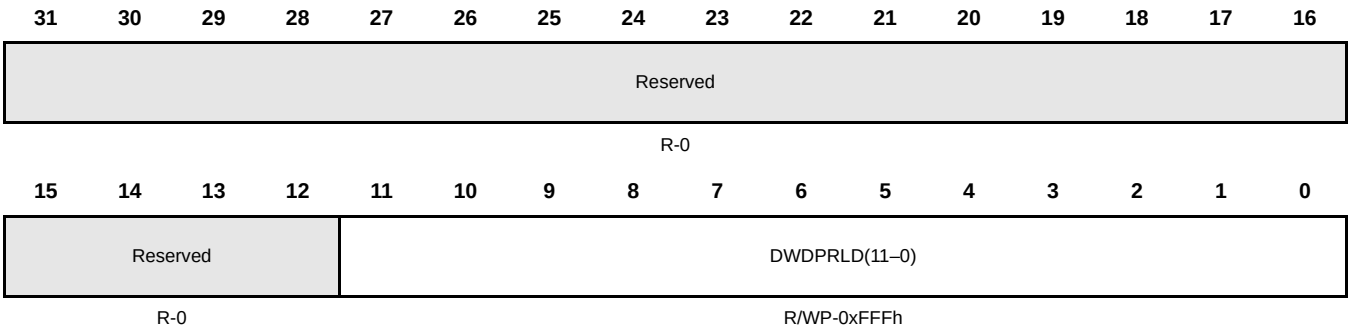
Table 16-25. Digital Watchdog Control Register (RTIDWDCTRL) Field Descriptions

Bit	Name	Value	Description
31-0	DWDCTRL		Digital Watchdog Control.
		0x5312ACED	*Read:* DWD counter is disabled. *Write:* State of DWD counter is unchanged (stays enabled or disabled).
		0xA98559DA	*Read:* DWD counter is enabled. *Write:* DWD counter is enabled
		Any other value	*Read: DWD counter state is unchanged (enabled or disabled).* *Write:* State of DWD counter is unchanged (stays enabled or disabled). **NOTE:** The RTIDWDCTRL register implements a one-write functionality, such that the application cannot write to this registermore than once. Writing the default value will not enable the watchdog as described above. Writing the enable value will start the watchdog counters. A write to RTIDWDCTRL will only be enabled after a system reset again.

16.3.26 RTI Digital Watchdog Preload Register (RTIDWDPRLD)

This register sets the expiration time of the DWD. This register is shown in Figure 16-30 and described in Table 16-26.

Figure 16-30. RTI Watchdog Preload Register (RTIDWDPRLD) [offset = 0x94]



R = Read, WP = Write in privileged mode only, -*n* = Value after reset

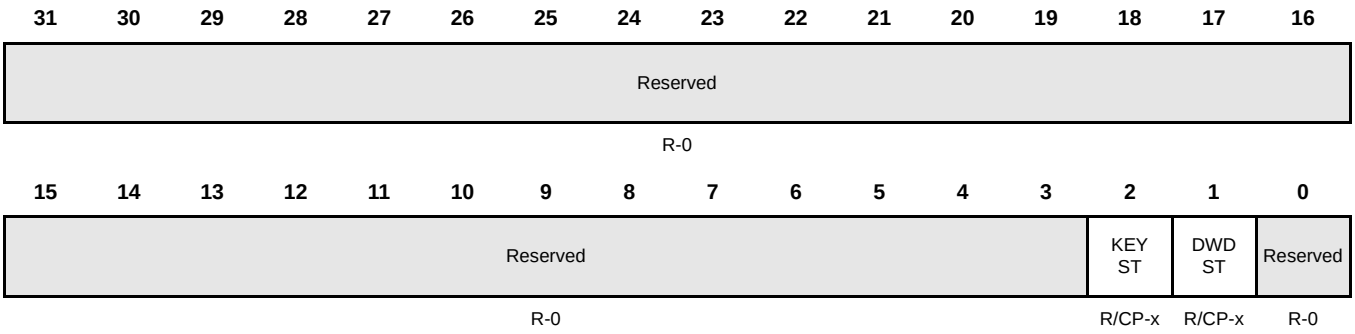
Table 16-26. RTI Watchdog Preload Register (RTIDWDPRLD) Field Descriptions

Bit	Name	Value	Description
31–12	Reserved		Reads return 0 and writes have no effect.
11–0	DWDPRLD(11–0)	0–FFFh	DWD preload value. *Read:* The current preload value is returned. *Write:* Set the preload value. The expiration time of the counter can be calculated with the formula: t_{exp} = (RTIDWDPRLD + 1) x 2^{13}/RTICLK where RTIDWDPRLD = 0...4095

16.3.27 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 reset. Only the user can clear these bits. These bits are only intended for debug purposes. This register is shown in Figure 16-31 and described in Table 16-27.

Figure 16-31. RTI Watchdog Status Register (RTIDWDSTATUS) [offset = 0x98]



R = Read, CP = Clear in privileged mode only, -*n* = Value after reset, -*x* = indeterminate

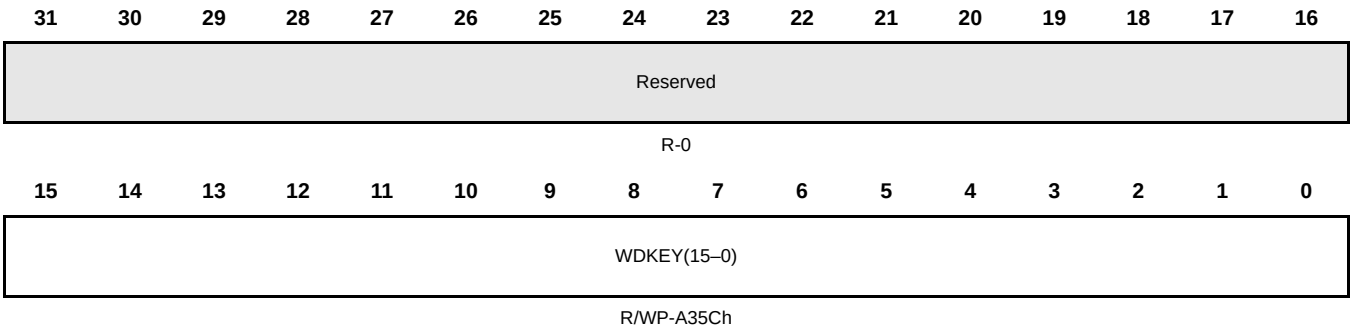
Table 16-27. RTI Watchdog Status Register (RTIDWDSTATUS) Field Descriptions

Bit	Name	Value	Description
31–3	Reserved		Reads return 0 and writes have no effect.
2	KEYST		Watchdog key status. This bit indicates a reset generated by a wrong key or key sequence written to the RTIWDKEY register.
		0	*Read:* No wrong key or key-sequence written. *Write:* The bit is unchanged.
		1	*Read:* wrong key or key-sequence written to RTIWDKEY register. *Write:* The bit is set to 0.
1	DWDST		DWD status.
		0	*Read:* No reset was generated. *Write:* The bit is unchanged.
		1	*Read:* A reset was generated. *Write:* The bit is set to 0.
0	Reserved		Reads return 0 and writes have no effect.

16.3.28 RTI Watchdog Key Register (RTIWDKEY)

This register must be written with the correct written key values to discharge the external capacitor. This register is shown in Figure 16-32 and described in Table 16-28.

Figure 16-32. RTI Watchdog Key Register (RTIDWDKEY) [offset = 0x9C]



R = Read, WP = Write in privileged mode only, -*n* = Value after reset

Table 16-28. RTI Watchdog Key Register (RTIDWDKEY) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Reads return 0 and writes have no effect.
15–0	WDKEY(15–0)	0–FFFFh	Watchdog key. These bits provide the key sequence location. Reads are indeterminate. A write of 0xE51A followed by 0xA35C in two separate write operations defines the key sequence and reloads the DWD. Writing any other value causes a reset, as shown in Table 16-29.

Note:
It has to be taken into account that the write to the RTIWDKEY register takes 3 VCLK cycles.

Table 16-29. Example of a WDKEY Sequence

Step	Value Written to WDKEY	Result
1	0x0A35C	No action
2	0x0A35C	No action
3	0x0E51A	WDKEY is enabled for reset by next 0x0A35C.
4	0x0E51A	WDKEY is enabled for reset by next 0x0A35C.
5	0x0E51A	WDKEY is enabled for reset by next 0x0A35C.
6	0x0A35C	Watchdog is reset.
7	0x0A35C	No action
8	0x0E51A	WDKEY is enabled for reset by next 0x0A35C.
9	0x0A35C	Watchdog is reset.
10	0x0E51A	WDKEY is enabled for reset by next 0x0A35C.
11	0x02345	System reset; incorrect value written to WDKEY.

16.3.29 RTI Digital Watchdog Down Counter (RTIDWDCNTR)

This register provides the current value of the DWD down counter. This register is shown in Figure 16-33 and described in Table 16-30.

Figure 16-33. RTI Watchdog Down Counter Register (RTIDWDCNTR) [offset = 0xA0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved							DWDCNTR(24–16)								
R-0							R-1FFh								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DWDCNTR(15–0)															
R-FFFFh															

R = Read, -*n* = Value after reset

Table 16-30. RTI Watchdog Down Counter Register (RTIDWDCNTR) Field Descriptions

Bit	Name	Value	Description
31–25	Reserved		Reads return 0 and writes have no effect.
24–0	DWDCNTR(24–0)	0–1FF FFFFh	DWD down counter. Reads return the current counter value.

Error Signaling (ESM) Module

This chapter provides the specification for the Error Signaling Module (ESM) that can be used to indicate a severe device failure on an external signal. The error indication is generated in addition to an error interrupt when a failure is recognized.

Topic **Page**

17.1 General Description. 898
17.2 Block diagram 899
17.3 Register Mapping. 900
17.4 Module Operation 901
17.5 Control Registers. 903

17.1 General Description

The Error Signaling Module (ESM) can indicate a severe device failure via an external pin.

The error pin is normally used as a second indication path to switch off the CPU by an external hardware device. Therefore the external device is able to reset the controller or keep the system in a fail-safe state by disabling the peripherals outside of the ECU.

17.1.1 Feature List

- Up to 64 interrupt/error channels are supported and configurable with up to 3 groups" (see device datasheet for device specific implementation).
  - 32 Channels with configurable interrupt and error pin behavior
  - 32 Error channels with predefined interrupt and error pin behavior
  - 32 Channels with predefined error pin behavior
- Scalable by design
- Error pin to signal severe device failure
- Configurable timebase for error signal
- Error forcing capability

Note:
Not all devices implement a dedicated $\overline{\text{ERROR}}$ pin. Please see the device specific data sheet for additional information.

17.2 Block diagram

Figure 17-1. Block Diagram

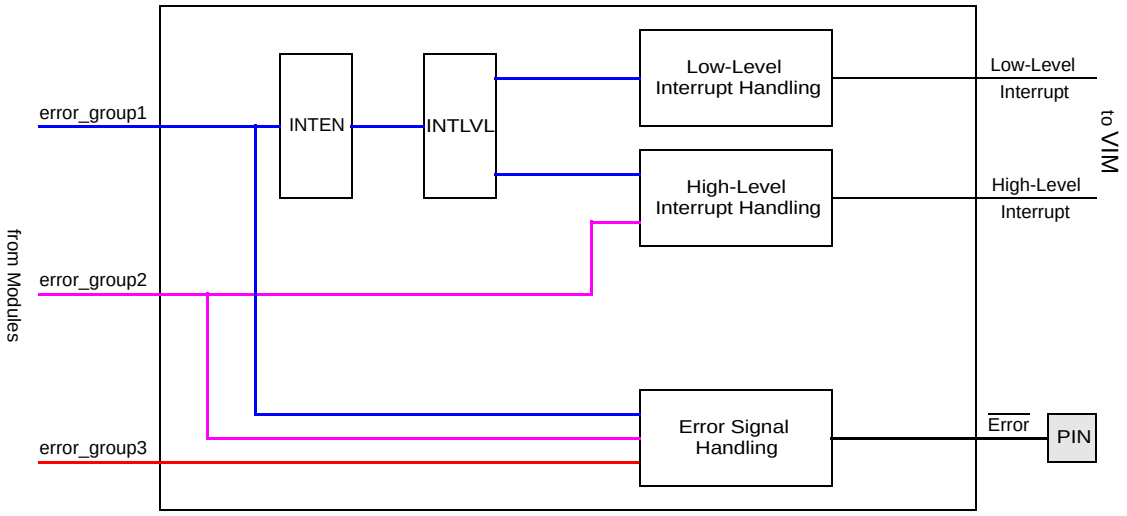
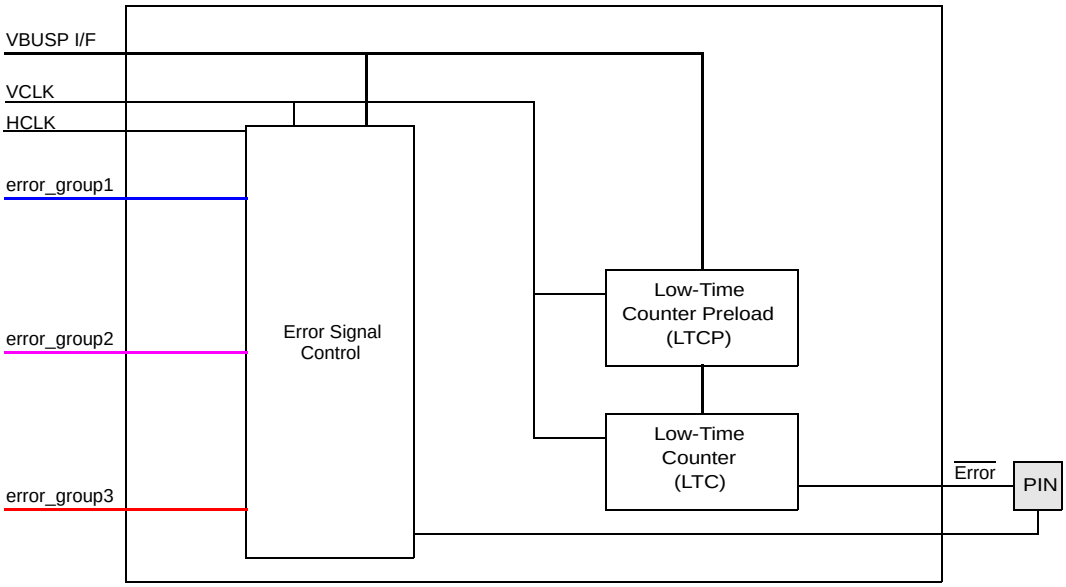


Figure 17-2. Error Signal Handling



17.3 Register Mapping

Table 17-1. Register Mapping

Offset Address†	Mnemonic	Name	Description	Page
0xFFFF_F500	ESMIEPSR1	ESM Influence Error Pin Set Register 1	Sets/shows Influence on Error Pin, Error Group 1	904
0xFFFF_F504	ESMIEPCR1	ESM Influence Error Pin Clear Register 1	Clears/shows Influence on Error Pin, Error Group 1	906
0xFFFF_F508	ESMIESR1	ESM Interrupt Enable Set Register 1	Sets/shows Interrupt Enable, Error Group 1	907
0xFFFF_F50C	ESMIECR1	ESM Interrupt Enable Clear Register 1	Clears/shows Interrupt Enable, Error Group 1	908
0xFFFF_F510	ESMILSR1	ESM Interrupt Level Set Register 1	Sets/shows Interrupt Level, Error Group 1	909
0xFFFF_F514	ESMILCR1	ESM Interrupt Level Clear Register 1	Clears/shows Interrupt Level, Error Group 1	910
0xFFFF_F518	ESMSR1	ESM Status Register 1	Shows Interrupt/Error Status Flag, Error Group 1	911
0xFFFF_F51C	ESMSR2	ESM Status Register 2	Shows Interrupt/Error Status Flag, Error Group 2	912
0xFFFF_F520	ESMSR3	ESM Status Register 3	Shows Interrupt/Error Status Flag, Error Group 3	913
0xFFFF_F524	ESMEPSR	ESM Error Pin Status Register	Shows Status of Error Pin	914
0xFFFF_F528	ESMIOFFHR	ESM Interrupt Offset High Register	Offset for High Level Interrupt	915
0xFFFF_F52C	ESMIOFFLR	ESM Interrupt Offset Low Register	Offset for Low Level Interrupt	916
0xFFFF_F530	ESMLTCR	ESM Low-Time Counter Register	Low Time Counter (triggered by peripheral clock)	917
0xFFFF_F534	ESMLTCPR	ESM Low-Time Counter Preload Register	Low Time Counter Preload Value	918
0xFFFF_F538	ESMEKR	ESM Error Key Register	Resets Error Pin or forces error on Error Pin for testability	919
0xFFFF_F53C	ESMSSR2	ESM Status Shadow Register 2	Shadow Register for ESMSR2, Error Group 2	920

17.4 Module Operation

For diagnosis reason it has to be possible to identify the source of each detected failure. The information about the source of the failure should be kept until power-on-reset. For several failures, a more detailed information (e.g. which addresses are affected) should be provided; this has to be supported by the module of the failure source. It must be possible to clear any failure information by software. The enabling or disabling functionality should be provided directly at the source of the failure indication and only for sources which are not considered “critical“. In case of “critical” failures, the interrupt should not be maskable as a target; the minimum requirement is a high level interrupt. It must be provided, that any other device checking function (e.g. clock monitoring) can also activate the error pin if enabled by software configuration.

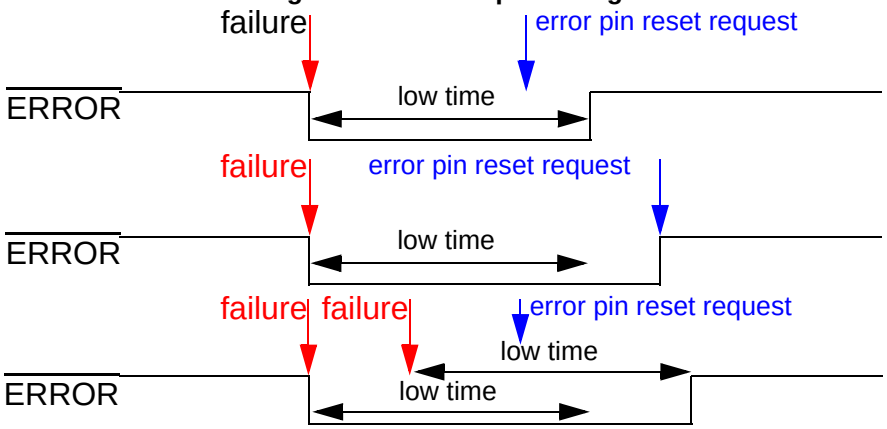
The pin must be low active; it is high as long as no failure is detected. The state of the pin must be readable by software. An internal reset must not affect the state of the pin. During power on reset the pin should have high impedance. A failure is latched by the ESM using HCLK. If a failure occurs the error pin changes it’s status to low in the next VCLK cycle. The pin should remain in this state for the time specified by the Low-Time Counter Preload register (LTCPR). Based on the period of the peripheral clock (VCLK), the total active time of the error pin ($\overline{\text{ERROR}}$) can be calculated as

$$t_{ERROR_low} = t_{VCLK} * (LTCP + 1) \quad \text{(EQ 1)}$$

If this period expires, the error pin must be set to high in case the reset of the error pin was requested by the software during the error pin low time by writing an appropriate key to the key register. If another failure occurs within this time the pin stays low and the low time counter must be reset. The detection mechanisms must not be disabled. (see Figure 17-3)

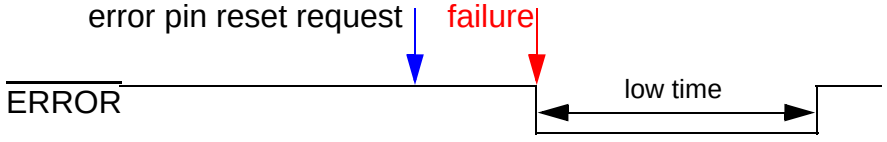
Note:
On the TMS470M Series device, this ERROR pin is not available.

Figure 17-3. error pin timing



If the error low time period expires, the error pin must be set to high in case the reset of the error pin was requested by the software even before the error pin low time has started (Figure 17-4). This case is not recommended and should be avoided by the application.

Figure 17-4. error pin timing



The above mentioned functionality must be testable by software in a way that a failure can be forced. This allows testing not only the error pin functionality but also the signal reaching the error pin activation unit. By writing a dedicated key to the error forcing key register the error pin is set to low for the specified time. If the error pin is low it must not be possible to set the error pin to high by software.

If the module is in error forcing mode, the application software should be aware of this, since it has requested the test, and therefore it can not handle the normal error functionality during this time. When a failure occurs while in error forcing mode, it gets still latched and the LTC is reset and stopped; the error output pin signal already shows the "low". When the mode is forced back to functional mode the LTC gets active. In case of pending failures, the error output pin will show “low” timed by the LTC, otherwise it will show “high” immediately.

As long as a failure is reported in functional mode (the error output pin is "low" for the LTC time) the module will not switch to the error forcing mode. The requested switching to error forcing mode is ignored. Therefore the application software should check for pending failures before it requests the error forcing mode.

The generating of interrupts and the activation of the error pin is shown in Table 17-2 below:

Table 17-2. ESM interrupt and error pin behavior

Error Group	Interrupt, Level	Influence on Error Pin
1	maskable, low/high	configurable
2	non-maskable, high	fix
3	none, ---	fix

Please refer to the specific device datasheet for implementation details.

The error pin gets into “output in-active” (high impedance) state with pull-down enabled during power-on-reset and into “output active” state with pull-down disabled during any other reset.

17.5 Control Registers

17.5.1 Control registers

This section describes the ESM registers. Each register begins on a word boundary. The registers support 32-bit, 16-bit and 8-bit accesses. The offset is relative to the associated peripheral select. The base address for the control registers is 0xFFFF F500.

Table 17-3. Module 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
0xFFFF_F500 ESMIEPSR1 Page 904	IEPSET131	IEPSET130	IEPSET29	IEPSET28	IEPSET27	IEPSET26	IEPSET25	IEPSET24	IEPSET23	IEPSET22	IEPSET21	IEPSET20	IEPSET19	IEPSET18	IEPSE17	IEPSET16
	IEPSET15	IEPSET14	IEPSET13	IEPSET12	IEPSET11	IEPSET10	IEPSET9	IEPSET8	IEPSET7	IEPSET6	IEPSET5	IEPSET4	IEPSET3	IEPSET2	IEPSET1	IEPSET0
0xFFFF_F504 ESMIEPCR1 Page 906	IEPCLR31	IEPCLR30	IEPCLR29	IEPCLR28	IEPCLR27	IEPCLR26	IEPCLR25	IEPCLR24	IEPCLR23	IEPCLR22	IEPCLR21	IEPCLR20	IEPCLR19	IEPCLR18	IEPCLR17	IEPCLR16
	IEPCLR15	IEPCLR14	IEPCLR13	IEPCLR12	IEPCLR11	IEPCLR10	IEPCLR9	IEPCLR8	IEPCLR7	IEPCLR6	IEPCLR5	IEPCLR4	IEPCLR3	IEPCLR2	IEPCLR1	IEPCLR0
0xFFFF_F508 ESMIESR1 Page 907	INTENSET31	INTENSET30	INTENSET29	INTENSET28	INTENSET27	INTENSET26	INTENSET25	INTENSET24	INTENSET23	INTENSET22	INTENSET21	INTENSET20	INTENSET19	INTENSET18	INTENSET17	INTENSET16
	INTENSET15	INTENSET14	INTENSET13	INTENSET12	INTENSET11	INTENSET10	INTENSET9	INTENSET8	INTENSET7	INTENSET6	INTENSET5	INTENSET4	INTENSET3	INTENSET2	INTENSET1	INTENSET0
0xFFFF_F50C ESMIECR1 Page 908	INTENCLR31	INTENCLR30	INTENCLR29	INTENCLR28	INTENCLR27	INTENCLR26	INTENCLR25	INTENCLR24	INTENCLR23	INTENCLR22	INTENCLR21	INTENCLR20	INTENCLR19	INTENCLR18	INTENCLR17	INTENCLR16
	INTENCLR15	INTENCLR14	INTENCLR13	INTENCLR12	INTENCLR11	INTENCLR10	INTENCLR9	INTENCLR8	INTENCLR7	INTENCLR6	INTENCLR5	INTENCLR4	INTENCLR3	INTENCLR2	INTENCLR1	INTENCLR0
0xFFFF_F510 ESMILSR1 Page 909	INTLVLSET31	INTLVLSET30	INTLVLSET29	INTLVLSET28	INTLVLSET27	INTLVLSET26	INTLVLSET25	INTLVLSET24	INTLVLSET23	INTLVLSET22	INTLVLSET21	INTLVLSET20	INTLVLSET19	INTLVLSET18	INTLVLSET17	INTLVLSET16
	INTLVLSET15	INTLVLSET14	INTLVLSET13	INTLVLSET12	INTLVLSET11	INTLVLSET10	INTLVLSET9	INTLVLSET8	INTLVLSET7	INTLVLSET6	INTLVLSET5	INTLVLSET4	INTLVLSET3	INTLVLSET2	INTLVLSET1	INTLVLSET0
0xFFFF_F514 ESMILCR1 Page 910	INTLVLCLR31	INTLVLCLR30	INTLVLCLR29	INTLVLCLR28	INTLVLCLR27	INTLVLCLR26	INTLVLCLR25	INTLVLCLR24	INTLVLCLR23	INTLVLCLR22	INTLVLCLR21	INTLVLCLR20	INTLVLCLR19	INTLVLCLR18	INTLVLCLR17	INTLVLCLR16
	INTLVLCLR15	INTLVLCLR14	INTLVLCLR13	INTLVLCLR12	INTLVLCLR11	INTLVLCLR10	INTLVLCLR9	INTLVLCLR8	INTLVLCLR7	INTLVLCLR6	INTLVLCLR5	INTLVLCLR4	INTLVLCLR3	INTLVLCLR2	INTLVLCLR1	INTLVLCLR0
0xFFFF_F518 ESMSR1 Page 911	ESF31	ESF30	ESF29	ESF28	ESF27	ESF26	ESF25	ESF24	ESF23	ESF22	ESF21	ESF20	ESF19	ESF18	ESF17	ESF16
	ESF15	ESF14	ESF13	ESF12	ESF11	ESF10	ESF9	ESF8	ESF7	ESF6	ESF5	ESF4	ESF3	ESF2	ESF1	ESF0
0xFFFF_F51C ESMSR2 Page 912	ESF31	ESF30	ESF29	ESF28	ESF27	ESF26	ESF25	ESF24	ESF23	ESF22	ESF21	ESF20	ESF19	ESF18	ESF17	ESF16
	ESF15	ESF14	ESF13	ESF12	ESF11	ESF10	ESF9	ESF8	ESF7	ESF6	ESF5	ESF4	ESF3	ESF2	ESF1	ESF0

Table 17-3. Module Registers (Continued)

0xFFFF_F520
ESMSR3
Page 913

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ESF31	ESF30	ESF29	ESF28	ESF27	ESF26	ESF25	ESF24	ESF23	ESF22	ESF21	ESF20	ESF19	ESF18	ESF17	ESF16
ESF15	ESF14	ESF13	ESF12	ESF11	ESF10	ESF9	ESF8	ESF7	ESF6	ESF5	ESF4	ESF3	ESF2	ESF1	ESF0

0xFFFF_F524
ESMEPSR
Page 914

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

0xFFFF_F528
ESMIOFFHR
Page 915

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
Reserved															INTOFFH[6:0]

0xFFFF_F52C
ESMIOFFLR
Page 916

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
Reserved															INTOFFL[6:0]

0xFFFF_F530
ESMLTCR
Page 917

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
LTC[15:0]															

0xFFFF_F534
ESMLTCPR
Page 918

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
LTCP[15:0]															

0xFFFF_F538
ESMEKR
Page 919

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
Reserved															EKEY[3:0]

0xFFFF_F53C
ESMSSR2
Page 920

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ESF31	ESF30	ESF29	ESF28	ESF27	ESF26	ESF25	ESF24	ESF23	ESF22	ESF21	ESF20	ESF19	ESF18	ESF17	ESF16
ESF15	ESF14	ESF13	ESF12	ESF11	ESF10	ESF9	ESF8	ESF7	ESF6	ESF5	ESF4	ESF3	ESF2	ESF1	ESF0

17.5.2 ESM Influence Error Pin Set/Status Register 1 (ESMIEPSR1)

The base address for this register is 0xFFFF_F500.

Figure 17-5. ESM Influence Error Pin Set/Status Register 1 (ESMIEPSR1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IEPSET31	IEPSET30	IEPSET29	IEPSET28	IEPSET27	IEPSET26	IEPSET25	IEPSET24	IEPSET23	IEPSET22	IEPSET21	IEPSET20	IEPSET19	IEPSET18	IEPSET17	IEPSET16
RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IEPSET15	IEPSET14	IEPSET13	IEPSET12	IEPSET11	IEPSET10	IEPSET9	IEPSET8	IEPSET7	IEPSET6	IEPSET5	IEPSET4	IEPSET3	IEPSET2	IEPSET1	IEPSET0
RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0

R = Read, W = Write, P = Privileged mode; *-n* = Value after reset

Table 17-4. ESM Influence Error Pin Set/Status Register 1 (ESMIEPSR1)

Bit	Name	Value	Description
31-0	IEPSET		Set Influence on Error Pin
		0	**User and privileged mode:** Read: Failure on channel x has no influence on error pin. **Privileged mode:** Write: Leaves the bit and the corresponding clear bit in the ESMIEPCR1 register unchanged.
		1	**User and privileged mode:** Read: Failure on channel x has influence on error pin. **Privileged mode:** Write: Enables failure influence on error pin and sets the corresponding clear bit in the ESMIEPCR1 register.

17.5.3 ESM Influence Error Pin Clear/Status Register 1 (ESMIEPCR1)

The base address for this register is 0xFFFF_F504.

Figure 17-6. ESM Influence Error Pin Clear/Status Register 1 (ESMIEPCR1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IEPCLR31	IEPCLR30	IEPCLR29	IEPCLR28	IEPCLR27	IEPCLR26	IEPCLR25	IEPCLR24	IEPCLR23	IEPCLR22	IEPCLR21	IEPCLR20	IEPCLR19	IEPCLR18	IEPCLR17	IEPCLR16
RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IEPCLR15	IEPCLR14	IEPCLR13	IEPCLR12	IEPCLR11	IEPCLR10	IEPCLR9	IEPCLR8	IEPCLR7	IEPCLR6	IEPCLR5	IEPCLR4	IEPCLR3	IEPCLR2	IEPCLR1	IEPCLR0
RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0

R = Read, W = Write, P = Privileged mode; *-n* = Value after reset

Table 17-5. ESM Influence Error Pin Clear/Status Register 1 (ESMIEPCR1)

Bit	Name	Value	Description
31-0	IEPCLR		Clear Influence on Error Pin
		0	**User and privileged mode:** Read: Failure on channel x has no influence on error pin. **Privileged mode:** Write: Leaves the bit and the corresponding set bit in the ESMIEPSR1 register unchanged.
		1	**User and privileged mode:** Read: Failure on channel x has influence on error pin. **Privileged mode:** Write: Disables failure influence on error pin and clears the corresponding set bit in the ESMIEPSR1 register.

17.5.4 ESM Interrupt Enable Set/Status Register 1 (ESMIESR1)

The base address for this register is 0xFFFF_F508.

Figure 17-7. ESM Interrupt Enable Set/Status Register 1 (ESMIESR1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INTENSET31	INTENSET30	INTENSET29	INTENSET28	INTENSET27	INTENSET26	INTENSET25	INTENSET24	INTENSET23	INTENSET22	INTENSET21	INTENSET20	INTENSET19	INTENSET18	INTENSET17	INTENSET16
RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
INTENSET15	INTENSET14	INTENSET13	INTENSET12	INTENSET11	INTENSET10	INTENSET9	INTENSET8	INTENSET7	INTENSET6	INTENSET5	INTENSET4	INTENSET3	INTENSET2	INTENSET1	INTENSET0
RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0

R = Read, W = Write, P = Privileged mode; *-n* = Value after reset

Table 17-6. ESM Interrupt Enable Set/Status Register 1 (ESMIESR1)

Bit	Name	Value	Description
31-0	INTENSET		Set interrupt Enable
		0	**User and privileged mode:** Read: Interrupt is disabled. **Privileged mode:** Write: Leaves the bit and the corresponding clear bit in the ESMIECR1 register unchanged.
		1	**User and privileged mode:** Read: Interrupt is enabled. **Privileged mode:** Write: Enables interrupt and sets the corresponding clear bit in the ESMIECR1 register.

17.5.5 ESM Interrupt Enable Clear/Status Register 1 (ESMIECR1)

The base address for this register is 0xFFFF_F50C.

Figure 17-8. ESM Interrupt Enable Clear/Status Register 1 (ESMIECR1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INTENCLR31	INTENCLR30	INTENCLR29	INTENCLR28	INTENCLR27	INTENCLR26	INTENCLR25	INTENCLR24	INTENCLR23	INTENCLR22	INTENCLR21	INTENCLR20	INTENCLR19	INTENCLR18	INTENCLR17	INTENCLR16
RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTENCLR15	INTENCLR14	INTENCLR13	INTENCLR12	INTENCLR11	INTENCLR10	INTENCLR9	INTENCLR8	INTENCLR7	INTENCLR6	INTENCLR5	INTENCLR4	INTENCLR3	INTENCLR2	INTENCLR1	INTENCLR0
RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0

R = Read, W = Write, P = Privileged mode; *-n* = Value after reset

Table 17-7. ESM Interrupt Enable Clear/Status Register 1 (ESMIECR1)

Bit	Name	Value	Description
31-0	INTENCLR		Clear interrupt Enable
		0	**User and privileged mode:** Read: Interrupt is disabled. **Privileged mode:** Write: Leaves the bit and the corresponding set bit in the ESMIESR1 register unchanged.
		1	**User and privileged mode:** Read: Interrupt is enabled. **Privileged mode:** Write: Disables interrupt and clears the corresponding set bit in the ESMIESR1 register.

17.5.6 ESM Interrupt Level Set/Status Register 1 (ESMILSR1)

The base address for this register is 0xFFFF_F510.

Figure 17-9. ESM Interrupt Level Set/Status Register *1 (ESMILSR1)*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INTLVLSET31	INTLVLSET30	INTLVLSET29	INTLVLSET28	INTLVLSET27	INTLVLSET26	INTLVLSET25	INTLVLSET24	INTLVLSET23	INTLVLSET22	INTLVLSET21	INTLVLSET20	INTLVLSET19	INTLVLSET18	INTLVLSET17	INTLVLSET16
RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
INTVLVSET15	INTVLVSET14	INTVLVSET13	INTVLVSET12	INTVLVSET11	INTVLVSET10	INTVLVSET9	INTVLVSET8	INTVLVSET7	INTVLVSET6	INTVLVSET5	INTVLVSET4	INTVLVSET3	INTVLVSET2	INTVLVSET1	INTVLVSET0
RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0

R = Read, W = Write, P = Privileged mode; *-n* = Value after reset

Table 17-8. ESM Interrupt Level Set/Status Register 1 (ESMILSR1)

Bit	Name	Value	Description
31-0	INTLVLSET		Set Interrupt Level
		0	**User and privileged mode:** Read: Interrupt of channel x is mapped to low level interrupt line. **Privileged mode:** Write: Leaves the bit and the corresponding clear bit in the ESMILCR1 register unchanged.
		1	**User and privileged mode:** Read: Interrupt of channel x is mapped to high level interrupt line. **Privileged mode:** Write: Maps interrupt of channel x to high level interrupt line and sets the corresponding clear bit in the ESMILCR1 register.

17.5.7 ESM Interrupt Level Clear/Status Register 1 (ESMILCR1)

The base address for this register is 0xFFFF_F514.

Figure 17-10. ESM Interrupt Level Clear/Status Register *1 (ESMILCR1)*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INTLVLCLR31	INTLVLCLR30	INTLVLCLR29	INTLVLCLR28	INTLVLCLR27	INTLVLCLR26	INTLVLCLR25	INTLVLCLR24	INTLVLCLR23	INTLVLCLR22	INTLVLCLR21	INTLVLCLR20	INTLVLCLR19	INTLVLCLR18	INTLVLCLR17	INTLVLCLR16
RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
INTVLVCLR15	INTVLVCLR14	INTVLVCLR13	INTVLVCLR12	INTVLVCLR11	INTVLVCLR10	INTVLVCLR9	INTVLVCLR8	INTVLVCLR7	INTVLVCLR6	INTVLVCLR5	INTVLVCLR4	INTVLVCLR3	INTVLVCLR2	INTVLVCLR1	INTVLVCLR0
RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0

R = Read, W = Write, P = Privileged mode; *-n* = Value after reset

Table 17-9. ESM Interrupt Level Clear/Status Register 1 (ESMILCR1)

Bit	Name	Value	Description
31-0	INTLVLCLR		Clear Interrupt Level
		0	**User and privileged mode:** Read: Interrupt of channel x is mapped to low level interrupt line. **Privileged mode:** Write: Leaves the bit and the corresponding set bit in the ESMILSR1 register unchanged.
		1	**User and privileged mode:** Read: Interrupt of channel x is mapped to high level interrupt line. **Privileged mode:** Write: Maps interrupt of channel x to low level interrupt line and clears the corresponding set bit in the ESMILSR1 register.

17.5.8 ESM Status Register 1 (ESMSR1)

The base address for this register is 0xFFFF_F518.

Figure 17-11. ESM Status Register 1 *(ESMSR1)*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ESF31	ESF30	ESF29	ESF28	ESF27	ESF26	ESF25	ESF24	ESF23	ESF22	ESF21	ESF20	ESF19	ESF18	ESF17	ESF16
RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ESF15	ESF14	ESF13	ESF12	ESF11	ESF10	ESF9	ESF8	ESF7	ESF6	ESF5	ESF4	ESF3	ESF2	ESF1	ESF0
RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0

R = Read, C = Clear, P = Privileged mode; *-n* = Value after reset/PORRST, *X* = Value unchanged

Table 17-10. ESM Status Register 1 (ESMSR1)

Bit	Name	Value	Description
31-0	ESF		Error Status Flag Provides status information on a pending error
		0	**User and privileged mode:** Read: No error occurred; no interrupt is pending **Privileged mode:** Write: Leaves the bit unchanged.
		1	**User and privileged mode:** Read: Error occurred; interrupt is pending. **Privileged mode:** Write: Clears the bit.

17.5.9 ESM Status Register 2 (ESMSR2)

The base address for this register is 0xFFFF_F51C.

Figure 17-12. ESM Status Register 2 *(ESMSR2)*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ESF31	ESF30	ESF29	ESF28	ESF27	ESF26	ESF25	ESF24	ESF23	ESF22	ESF21	ESF20	ESF19	ESF18	ESF17	ESF16
RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ESF15	ESF14	ESF13	ESF12	ESF11	ESF10	ESF9	ESF8	ESF7	ESF6	ESF5	ESF4	ESF3	ESF2	ESF1	ESF0
RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0

R = Read, C = Clear, P = Privileged mode; *-n* = Value after reset

Table 17-11. ESM Status Register 2 (ESMSR2)

Bit	Name	Value	Description
31-0	ESF		Error Status Flag Provides status information on a pending error.
		0	**User and privileged mode:** Read: No error occurred; no interrupt is pending **Privileged mode:** Write: Leaves the bit unchanged.
		1	**User and privileged mode:** Read: Error occurred; interrupt is pending. **Privileged mode:** Write: Clears the bit.
			Note: In normal operation the flag gets cleared when reading the appropriate vector in the ESMIOFFHR offset register.

17.5.10 ESM Status Register 3 (ESMSR3)

The base address for this register is 0xFFFF_F520.

Figure 17-13. ESM Status Register 3 *(ESMSR3)*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ESF31	ESF30	ESF29	ESF28	ESF27	ESF26	ESF25	ESF24	ESF23	ESF22	ESF21	ESF20	ESF19	ESF18	ESF17	ESF16
RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ESF15	ESF14	ESF13	ESF12	ESF11	ESF10	ESF9	ESF8	ESF7	ESF6	ESF5	ESF4	ESF3	ESF2	ESF1	ESF0
RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0

R = Read, C = Clear, P = Privileged mode; *-n* = Value after reset/PORRST, *X* = Value unchanged

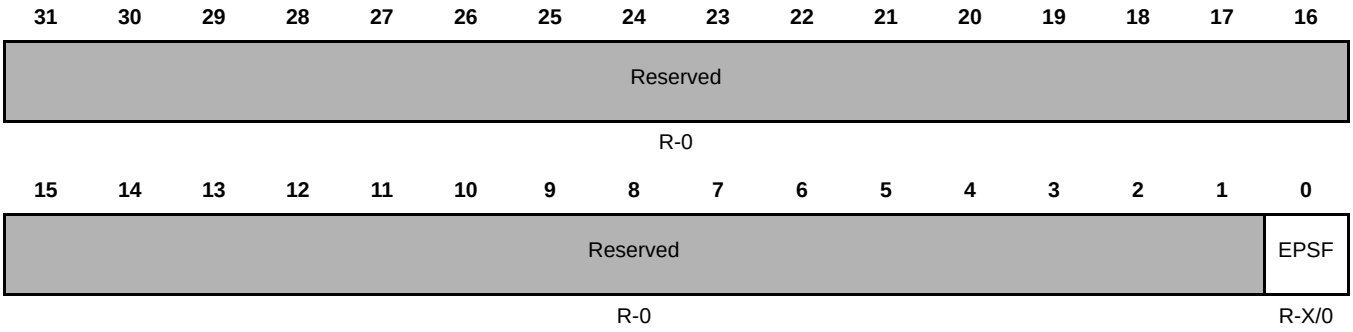
Table 17-12. ESM Status Register 3 (ESMSR3)

Bit	Name	Value	Description
31-0	ESF		Error Status Flag Provides status information on a pending error
		0	**User and privileged mode:** Read: No error occurred. **Privileged mode:** Write: Leaves the bit unchanged.
		1	**User and privileged mode:** Read: Error occurred. **Privileged mode:** Write: Clears the bit.

17.5.11 ESM Error Pin Status Register (ESMEPSR)

The base address for this register is 0xFFFF_F524.

Figure 17-14. ESM Error Pin Status Register (ESMEPSR)



R = Read; *-n* = Value after reset/PORRST, *X* = Value unchanged

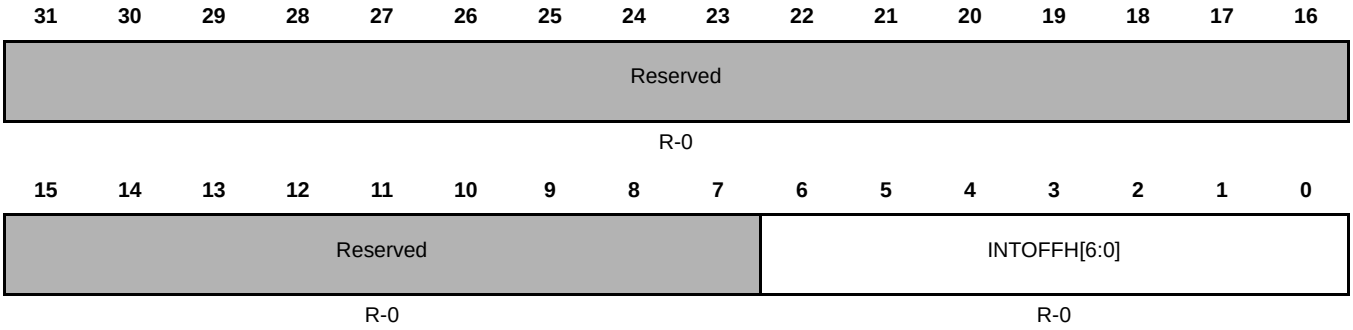
Table 17-13. ESM Error Pin Status Register (ESMEPSR)

Bit	Name	Value	Description
31–1	Reserved		These bits are read as zero, and writes have no effect.
0	EPSF		Error Pin Status Flag Provides status information for the Error Pin
		0	**User and privileged mode:** Read: Error Pin is low (active) if any error has occurred. Write: Writes have no effect.
		1	**User and privileged mode:** Read: Error Pin is high if no error has occurred. Write: Writes have no effect.

17.5.12 ESM Interrupt Offset High Register (ESMIOFFHR)

The base address for this register is 0xFFFF_F528.

Figure 17-15. ESM Interrupt Offset High Register *(ESMIOFFHR)*



R = Read; *-n* = Value after reset

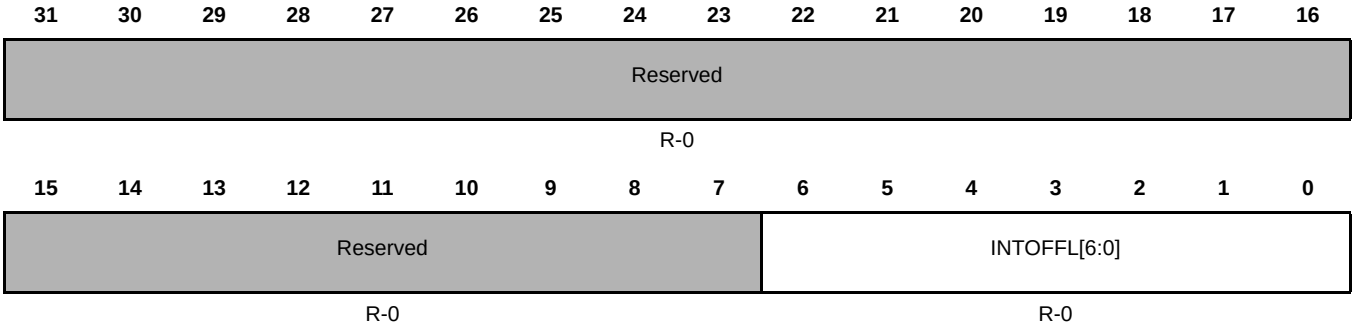
Table 17-14. ESM Interrupt Offset High Register (ESMIOFFHR)

Bit	Name	Value	Description
31–7	Reserved		These bits are read as zero, and 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 group 2 have higher priority than interrupts of error group 1. 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.
		0000000	No pending interrupt.
		0000001	Interrupt pending for channel 0, error group1.
		...	
		0100000	Interrupt pending for channel 31, error group1.
		0100001	Interrupt pending for channel 0, error group2.
		...	
		1000000	Interrupt pending for channel 31, error group2.
			Note: Reading the interrupt vector will clear the corresponding flag in the ESMSR2 register and the offset register gets updated.
			User and privileged mode (write):
			Writes have no effect.

17.5.13 ESM Interrupt Offset Low Register (ESMIOFFLR)

The base address for this register is 0xFFFF_F52C.

Figure 17-16. ESM Interrupt Offset Low Register *(ESMIOFFLR)*



R = Read; *-n* = Value after reset

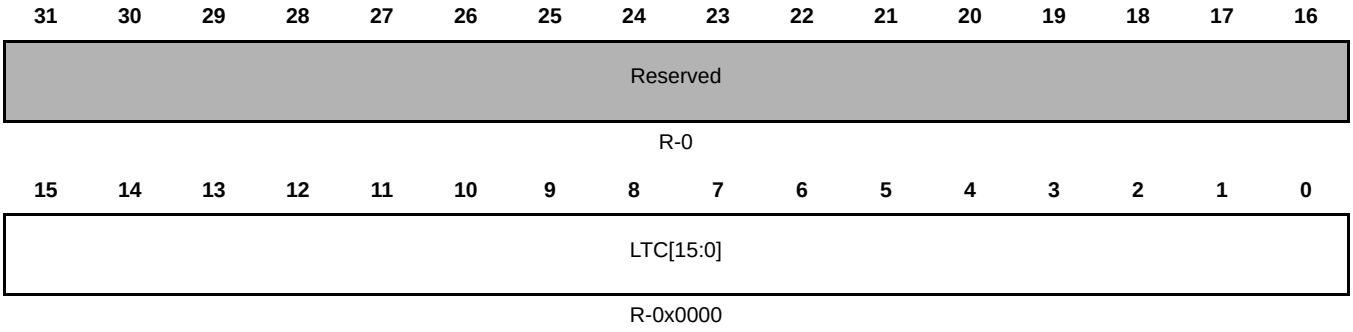
Table 17-15. ESM Interrupt Offset Low Register (ESMIOFFLR)

Bit	Name	Value	Description
31–7	Reserved		Reads return 0 and 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.
		0000000	No pending interrupt.
		0000001	Interrupt pending for channel 0, error group1.
		...	
		0100000	Interrupt pending for channel 31, error group1.
			Note: Group2 interrupts are fixed to the high level interrupt line only. **User and privileged mode (write):** Writes have no effect.

17.5.14 ESM Low-Time Counter Register (ESMLTCR)

The base address for this register is 0xFFFF_F530.

Figure 17-17. ESM Low-Time Counter Register *(ESMLTCR)*



R = Read; *-n* = Value after reset/PORRST

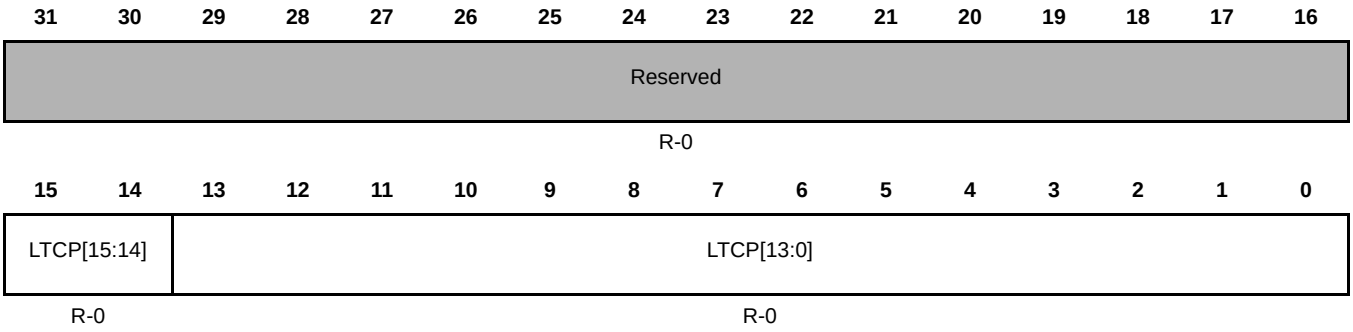
Table 17-16. ESM Low-Time Counter Register (ESMLTCR)

Bit	Name	Value	Description
31–16	Reserved		These bits are read as zero, and writes have no effect.
15-0	LTC		Error Pin Low-Time Counter 16bit pre-loadable down-counter to control low-time of error pin. The low-time counter is triggered by the peripheral clock (VCLK). **Note:** After reset the low time counter is set to the default pre-load value of the ESMLTCPR to ensure a minimum low time of 100us running at a maximum frequency of 100MHz.

17.5.15 ESM Low-Time Counter Preload Register (ESMLTCPR)

The base address for this register is 0xFFFF_F534.

Figure 17-18. ESM Low-Time Counter Preload Register *(ESMLTCPR)*



R = Read, W = Write, P = Privileged mode; *-n* = Value after reset

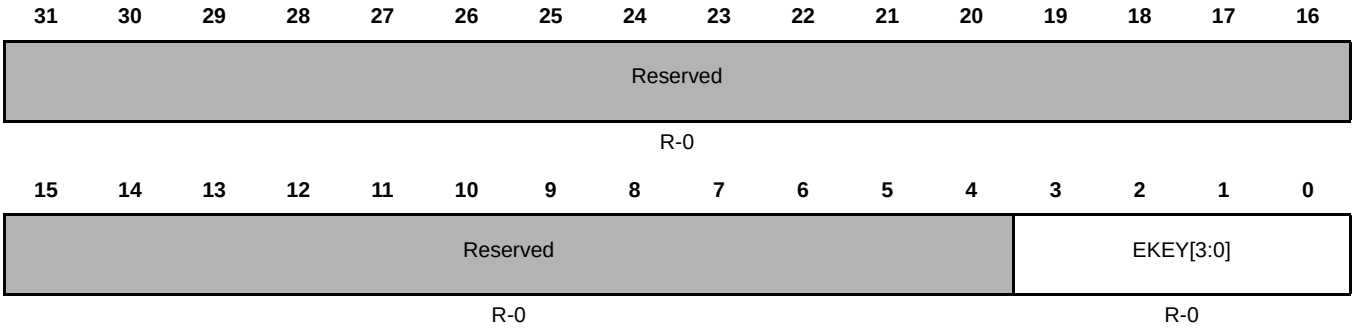
Table 17-17. ESM Low-Time Counter Preload Register (ESMLTCPR)

Bit	Name	Value	Description
31–16	Reserved		These bits are read as zero, and writes have no effect.
15-0	LTCP		Error Pin Low-Time Counter Pre-load Value 16bit pre-load value for the error pin low-time counter. **Note:** Only LTCP.15 and LTCP.14 are configurable (privileged mode write) to ensure a minimum low time of 100us running at a maximum frequency of 100MHz.

17.5.16 ESM Error Key Register (ESMEKR)

The base address for this register is 0xFFFF_F538.

Figure 17-19. ESM Error Key Register *(ESMEKR)*



R = Read, W = Write, P = Privileged mode; *-n* = Value after reset

Table 17-18. ESM Error Key Register (ESMEKR)

Bit	Name	Value	Description
31–4	Reserved		Reads return 0 and writes have no effect.
3-0	EKEY		Error Key The key to reset the error pin or to force an error on the error pin. **User and privileged mode (read):** Returns current value of the EKEY. **Privileged mode (write):**
		0000	Activates normal mode (recommended default mode).
		1010	Forces error on error pin.
		0101	Error pin set to high when the low time counter (LTC) has completed; then the ESMEKR.EKEY bit will switch back to normal mode (EKEY = 0000)
		all other values	Activates normal mode.

17.5.17 ESM Status Shadow Register 2 (ESMSSR2)

The base address for this register is 0xFFFF_F53C.

Figure 17-20. ESM Status Shadow Register 2 *(ESMSSR2)*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ESF31	ESF30	ESF29	ESF28	ESF27	ESF26	ESF25	ESF24	ESF23	ESF22	ESF21	ESF20	ESF19	ESF18	ESF17	ESF16
RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ESF15	ESF14	ESF13	ESF12	ESF11	ESF10	ESF9	ESF8	ESF7	ESF6	ESF5	ESF4	ESF3	ESF2	ESF1	ESF0
RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0

R = Read, C = Clear, P = Privileged mode; *-n* = Value after reset/PORRST, *X* = Value unchanged

Table 17-19. ESM Status Shadow Register 2 (ESMSSR2)

Bit	Name	Value	Description
31-0	ESF		Error Status Flag Shadow register for status information on pending error.
		0	**User and privileged mode:** Read: No error occurred. **Privileged mode:** Write: Leaves the bit unchanged.
		1	**User and privileged mode:** Read: Error occurred. **Privileged mode:** Write: Clears the bit.
			Note: Errors are stored until they are cleared by the software or at power-on reset (PORRST).

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 2.0 protocol specified in the *LIN Specification Package*. This module can be configured to operate in either SCI(UART) or LIN mode

Topic **Page**

18.1 Introduction and Features . . . 922
18.2 Block Diagram . . . 924
18.3 SCI Communication Formats . . . 927
18.4 SCI Interrupts . . . 935
18.5 SCI DMA Interface . . . 938
18.6 SCI Configurations . . . 939
18.7 SCI Low Power Mode . . . 941
18.8 LIN Communication Formats . . . 942
18.9 LIN Interrupts . . . 959
18.10 LIN DMA Interface . . . 960
18.11 LIN Configurations . . . 961
18.12 Low-Power Mode . . . 963
18.13 Emulation Mode . . . 965
18.14 SCI/LIN Control Registers . . . 966
18.15 GPIO Functionality . . . 1049

18.1 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 functinality 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 commom for LIN and SCI, the register descriptions has notes to identify the register / bit usage in different modes.

18.1.1 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.

18.1.2 LIN Features

The following are the features of the LIN module:

- Compatibility with LIN 1.3 or 2.0 protocols.
- LIN2.1 protocol Master Complaint.
- 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

18.2 Block Diagram

The SCI/LIN module contains core SCI block with added sub-blocks to support LIN protocol.

Three Major component 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 (interface clock generated by the system module).

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 18-1 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 multibuffered 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 18-2 shows the SCI/LIN block diagram.

Figure 18-1. Detailed SCI Block Diagram

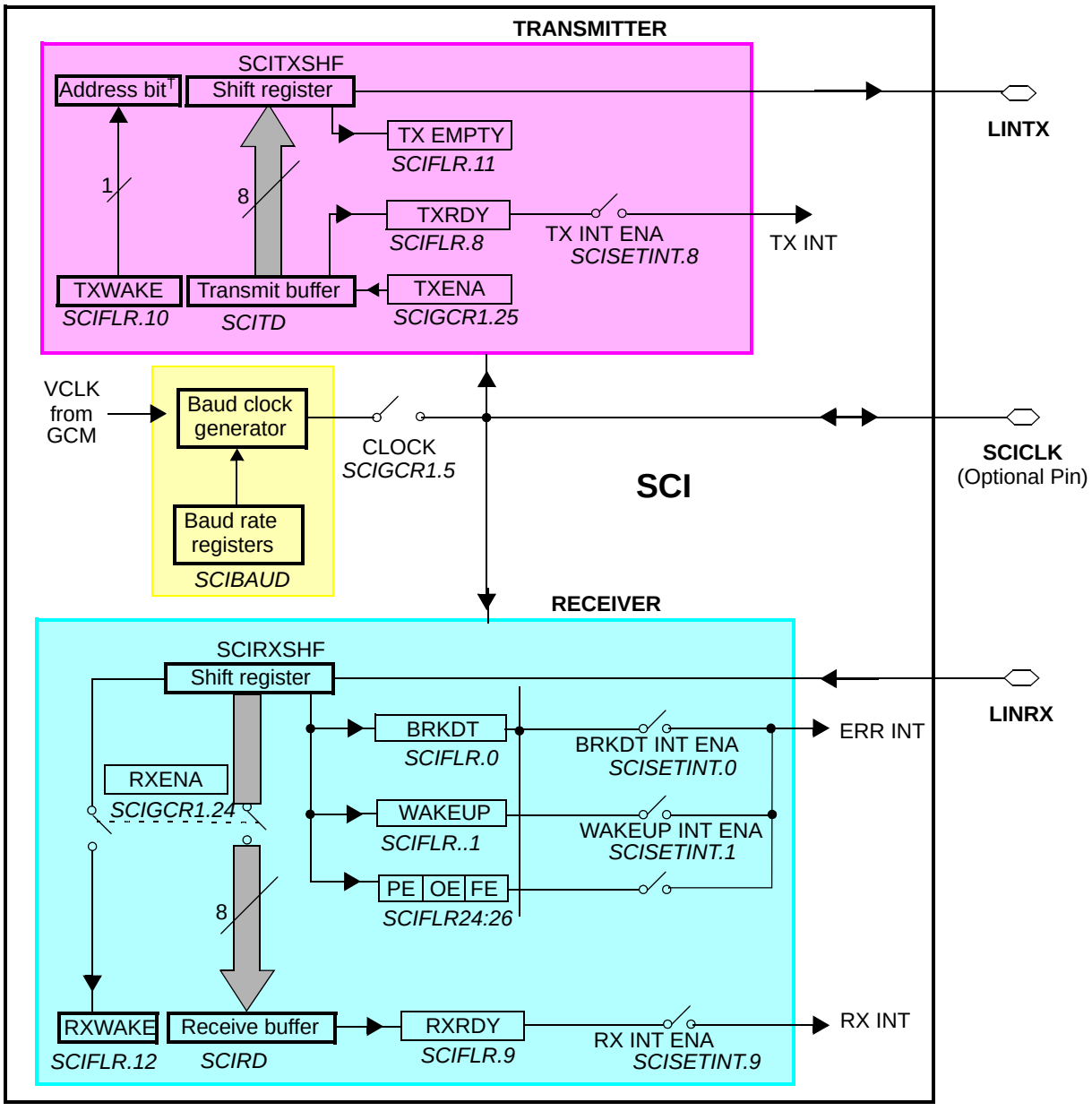
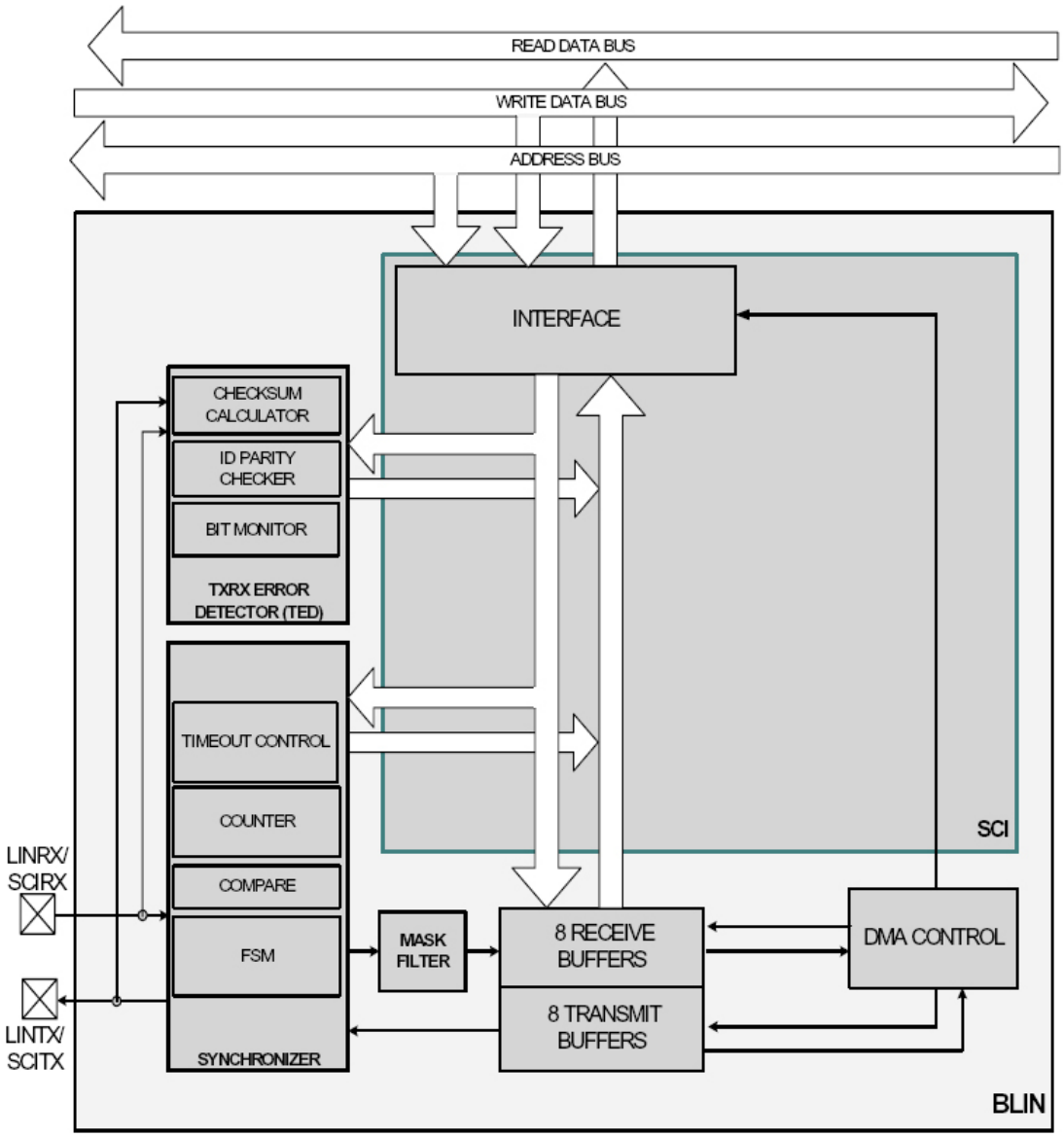


Figure 18-2. SCI/LIN Block Diagram



18.3 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

18.3.1 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

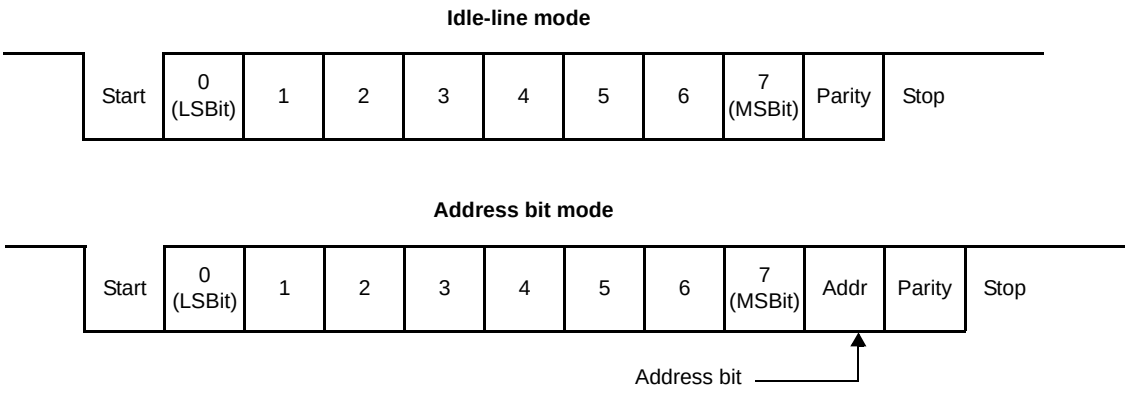
The frame format for both the transmitter and receiver is programmable through the bits in the SCIGCR1 register. Both receive and transmit data is in nonreturn to zero (NRZ) format, which means that the transmit and receive lines are at logic high when idle. Each frame transmission begins with a start 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 18-3.

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 18-3 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 18-3 use one stop bit per frame.

Figure 18-3. Typical SCI Data Frame Formats



18.3.2 SCI Timing Mode

The SCI can be configured to use asynchronous or isosynchronous timing using TIMING MODE bit in SCIGCR1 register.

18.3.2.1 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.

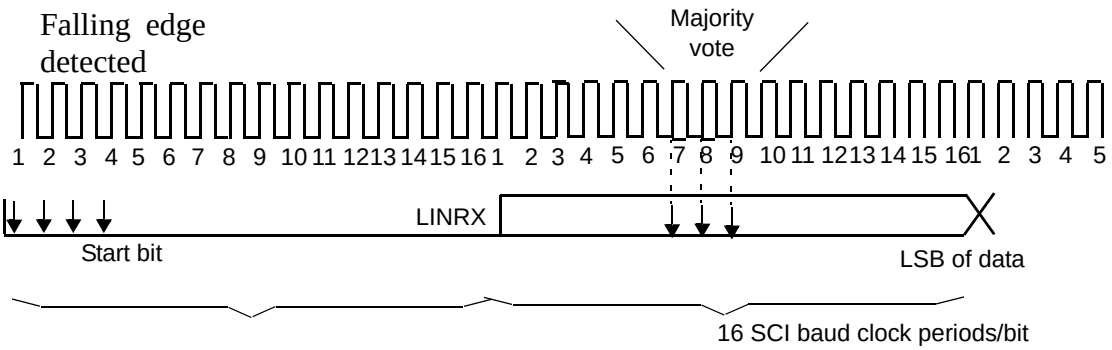
In the asynchronous timing mode, each bit in a frame has a duration of 16 SCI baud clock periods. Each bit therefore consists of 16 samples (one for each clock period). When the SCI is using asynchronous mode, the baud rates of all communicating devices must match as closely as possible. Receive errors 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 Stat 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 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 18-4 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 18-4. Asynchronous Communication Bit Timing



18.3.2.2 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.**

18.3.3 SCI Baud Rate

The SCI/LIN has an internally generated serial clock determined by the VCLK and the prescalers P and M in this register. The SCI uses the 24-bit integer prescaler P value of BRSR register to select the required baud rates. The additional 4-bit fractional divider M refines the baudrate selection.

In asynchronous timing mode, the SCI generates a baud clock according to the following formula:

$$SCICLK\ Frequency = \left| \frac{VCLK\ Frequency}{\left(P + 1 + \frac{M}{16}\right)} \right|$$

$$Asynchronous\ baud\ value = \left(\frac{SCICLK\ Frequency}{16}\right)$$

For P = 0,

$$Asynchronous\ baud\ value = \left(\frac{VCLK\ Frequency}{32}\right)$$

18.3.3.1 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 (BRSR[27:24]), the superfractional divider uses an additional 3-bit modulating value, illustrated in Table 18-2. 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 18-1), as shown in Table 18-2.

The baud rate will vary over a data field to average according to the BRSR[30:28] value by a "**d**" fraction of the peripheral internal clock: **0<d<1.** Refer Figure 18-5 for a simple Average "d' calculation based on "U" value(BRSR[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), $T^{i}_{bit} = \left[16\left(P + 1 + \frac{M}{16}\right) + d\right]T_{VCLK}$

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), $T^{a}_{bit} = \left[16\left(P + 1 + \frac{M}{16}\right) + d\right]T_{VCLK}$

For P= 0 $T_{bit} = 32T_{VCLK}$

.

Table 18-1. Superfractional Bit Modulation for SCI Mode (Normal Configuration)

BRS[30:28]	Start Bit	D[0]	D[1]	D[2]	D[3]	D[4]	D[5]	D[6]	D[7]	Stop Bit
0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	1	0
2	1	0	0	0	1	0	0	0	1	0
3	1	0	1	0	1	0	0	0	1	0
4	1	0	1	0	1	0	1	0	1	0
5	1	1	1	0	1	0	1	0	1	1
6	1	1	1	0	1	1	1	0	1	1
7	1	1	1	1	1	1	1	0	1	1

1 Normal configuration = Start + 8 Data Bits + Stop Bit

Table 18-2. Superfractional Bit Modulation for SCI Mode (Maximum Configuration)

BRS [30:28]	Start Bit	D[0]	D[1]	D[2]	D[3]	D[4]	D[5]	D[6]	D[7]	Addr	Parity	Stop 0	Stop 1
0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	1	0	0	0	0
2	1	0	0	0	1	0	0	0	1	0	0	0	1
3	1	0	1	0	1	0	0	0	1	0	1	0	1
4	1	0	1	0	1	0	1	0	1	0	1	0	1
5	1	1	1	0	1	0	1	0	1	1	1	0	1
6	1	1	1	0	1	1	1	0	1	1	1	0	1
7	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

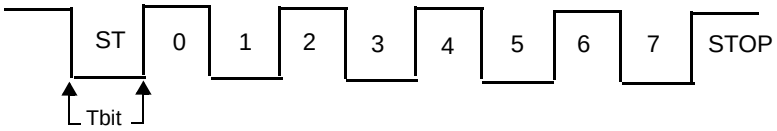
Table 18-3. SCI Mode (Minimum Configuration)

BRS[30:28]	Start Bit	D[0]	Stop
0	0	0	0
1	1	0	0
2	1	0	0
3	1	0	1
4	1	0	1
5	1	1	1
6	1	1	1
7	1	1	1

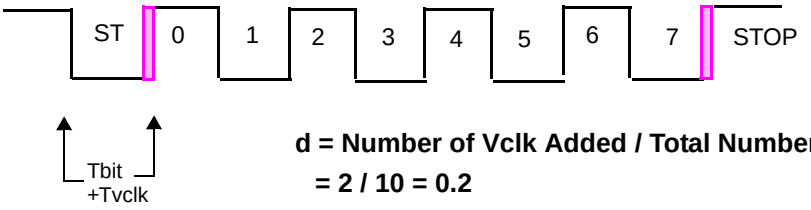
1 Minimum configuration = Start + 1 Data Bits + Stop Bit

Figure 18-5. Superfractional Divider Examplet

Normal Data Frame with BRS[31:28] = 0



Normal Data Frame with BRS[31:28] = 1 (Refer Table 18-1)



d = Number of Vclk Added / Total Number of Bits
= 2 / 10 = 0.2

18.3.4 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 Support two multi processor 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.

18.3.4.1 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 18-6 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:

Step1 : Write a 1 to the TXWAKE bit.

Step2 : 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.

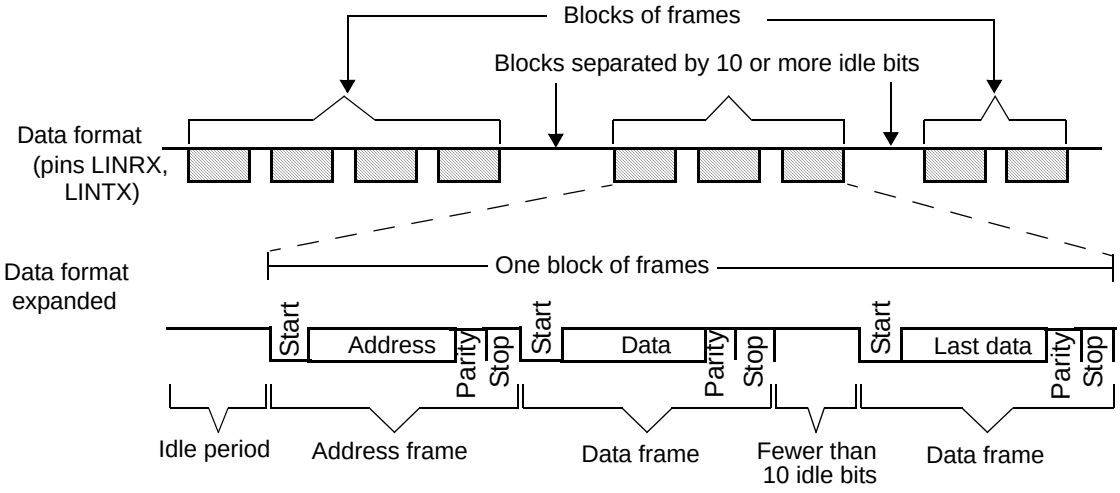
Step3 : Wait for the SCI to clear the TXWAKE flag.

Step4 : 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 18-6. Idle-Line Multiprocessor Communication Format



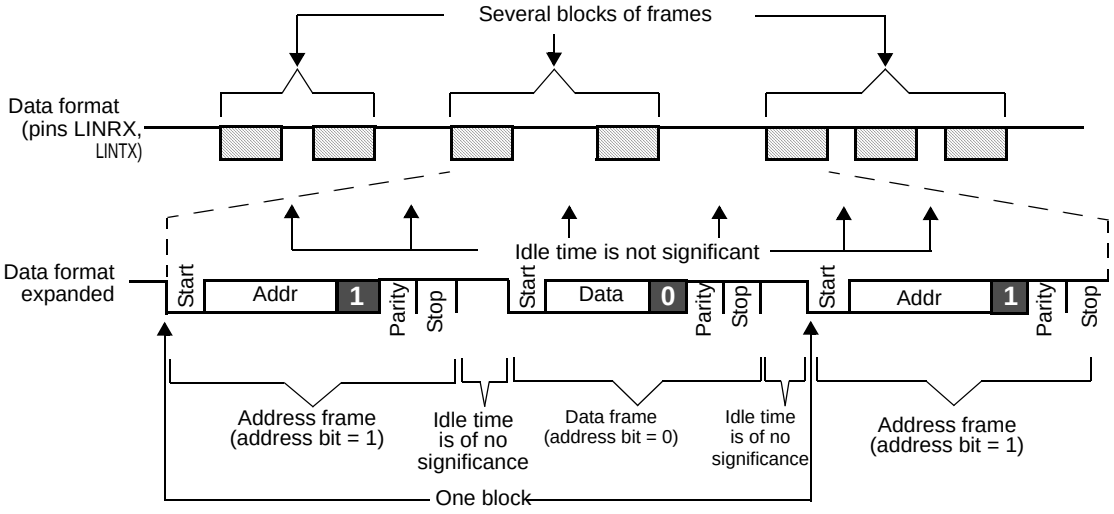
18.3.4.2 Address-Bit Multiprocessor Mode

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 18-7 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 18-7. Address-Bit Multiprocessor Communication Format



18.3.5 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 seperate Receive and transmit buffers. Multi buffered mode is enabled by setting the MBUF MODE bit.

The multibuffer 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 18-8 and Figure 18-9 shows the receive and transmit multibuffer functional block diagram.

Figure 18-8. Receive Buffers

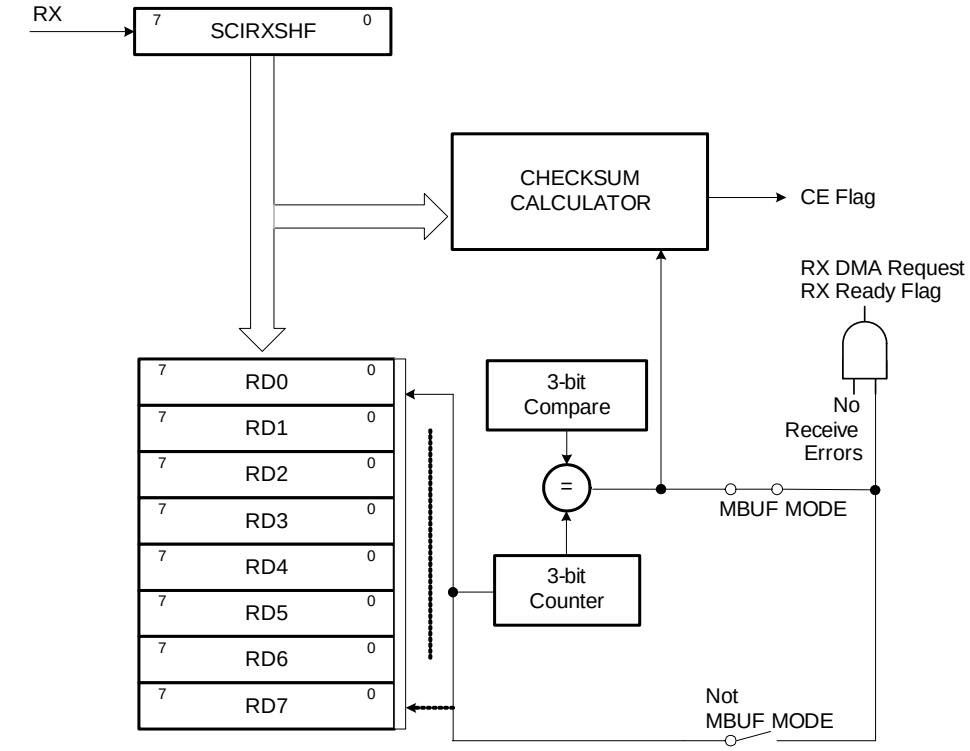
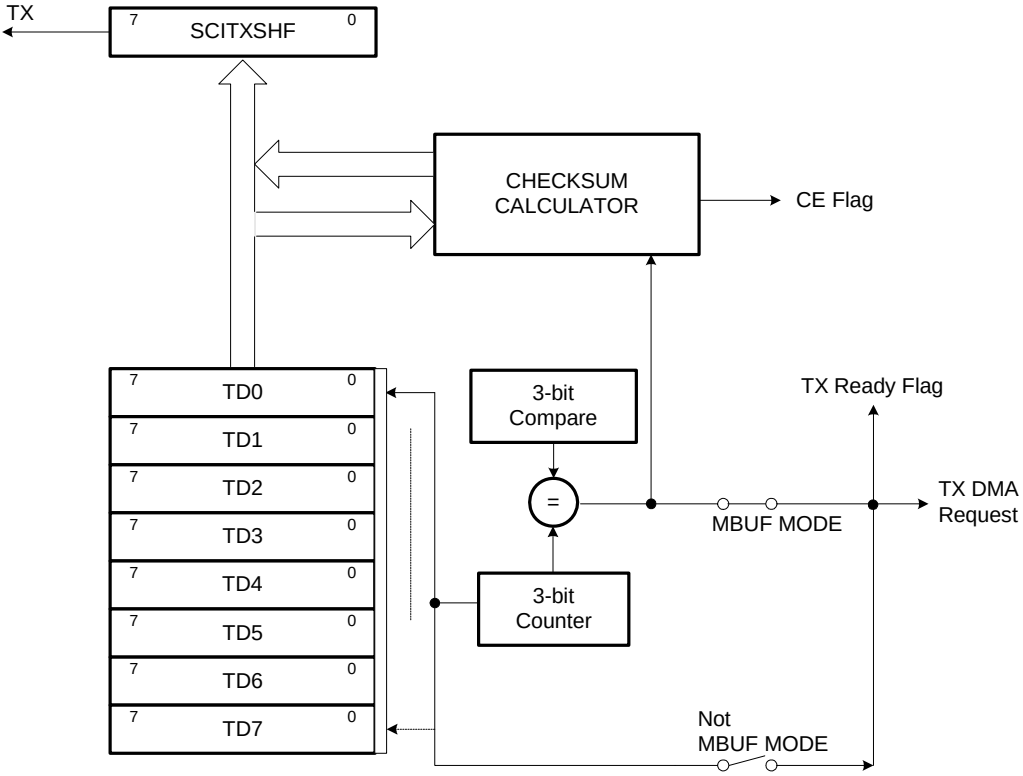


Figure 18-9. Transmit Buffers



18.4 SCI Interrupts

The SCI/LIN module has two interrupt lines, level 0 and level 1, to the vectored interrupt manager (VIM) module (see Figure 18-10). 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 18-10. General Interrupt Scheme

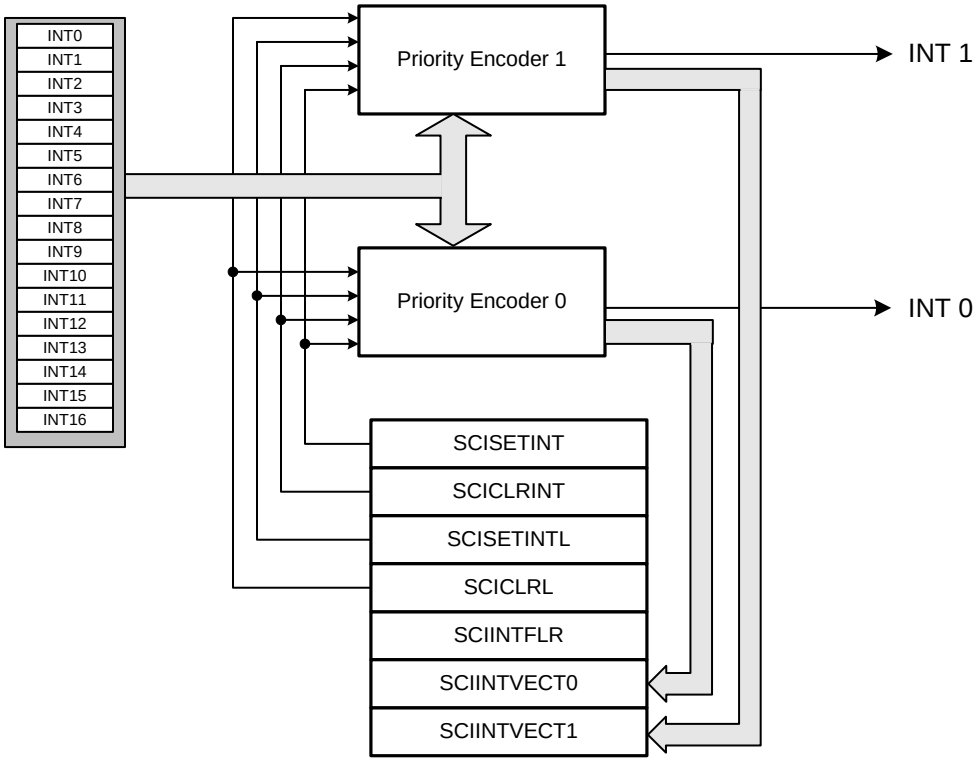
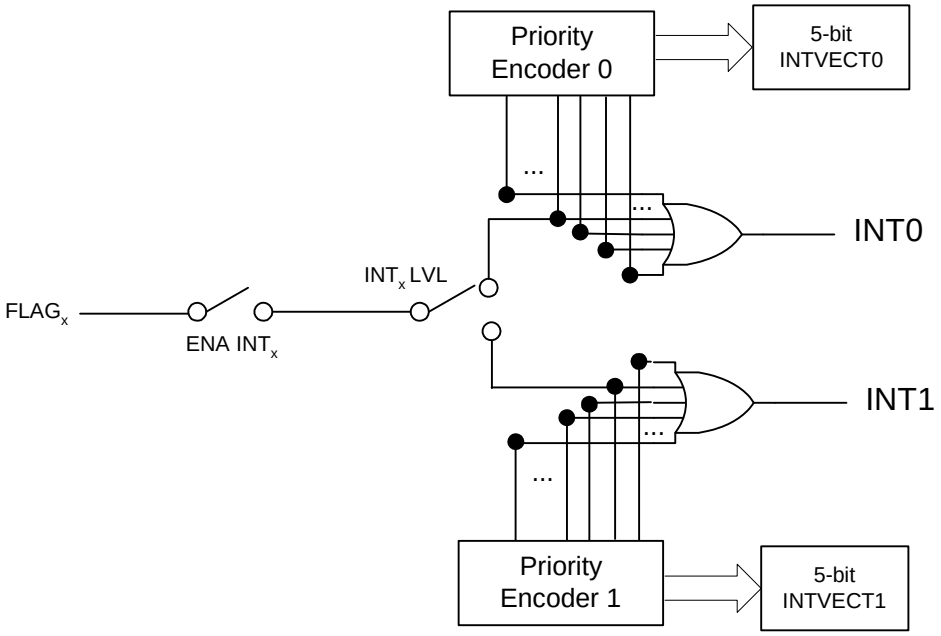


Figure 18-11. Interrupt Generation for Given Flags



18.4.1 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.

18.4.2 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 recived 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.

18.4.3 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 interupt is triggered once WAKEUP flag is set.

18.4.4 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)

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 18-4.

Table 18-4. 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.

18.5 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 multibuffer mode bit (MBUF MODE) is enabled or not. For DMA module configuration refer

18.5.0.1 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 sucessfully.

If the multibuffer option is disabled, then DMA requests will be generated on a byte-per-byte basis until all the expected response data fields are received.

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.

18.5.0.2 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 TXRDY bit is set or after a transmission of programmed no.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 multibuffer option is disabled, then DMA requests will be generated on a byte-per-byte basis until all bytes are transferred.

18.6 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 is set to 1. Of particular importance is the SWnRST bit. This active-low bit is 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 (one) 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 SWnRST is held low the entire time that the SCI is being configured, the order in which the registers are programmed is not important.

- Enable SCI by setting RESET bit.
- Clear SWnRST to 0 before configuring the SCI.
- Select the desired frame format by programming SCIGCR1.
- Configure the LINRX and LINTX pins for SCI functionality by setting the RX FUNC and TX FUNC bit.
- Select the baud rate to be used for communication by programming BRSR.
- Select internal clock by programming the CLOCK bit.
- Set the CONT bit to make SCI not to 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 to connect the transmitter to the receiver internally. (This feature is used to perform a self-test.)
- Select the receiver enable RXENA bit if data is to be received.
- Select the transmit enable TXENA bit if data is to be transmitted.
- Set SWnRST to 1 after the SCI is configured.
- Perform Receive or Transmit data. (see Section 18.6.1 / Section 18.6.2)

18.6.1 Receiving Data

The SCI receiver is enabled to receive messages if 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.

18.6.1.1 Receiving Data in Single-Buffer Mode

Single Buffer Mode is selected when MBUF MODE bit is 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 FE, OE, or PE 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.

User can receive data by

1) Polling Receive Ready Flag

2) Receive Interrupt

3) DMA

In polling method, software can poll for RXRDY bit and read the data from SCIRD register once RXRDY is set high. CPU is unnecessarly overloaded by selecting Polling mode. To avoid this user can use either

Interrupt or DMA method. To use interrupt method SET RX INT bit should be set and to use DMA SET RX DMA bit should be set. Either an Interrupt or a DMA request is generated the moment RXRDY is set.

18.6.1.2 Receiving Data in Multi-Buffer Mode

Multi-Buffer Mode is selected when MBUF MODE bit is 1. In this mode SCI sets the RXRDY bit when programmed number of data are received in the receive buffer, the complete frame. The error condition detection logic is same as Single Buffer Mode, except that it monitors for the complete frame. Like Single Buffer Mode the user can use either Interrupt, DMA or polling method to read the data. The received data has to be read from the LINRD0 and LINRD1 register, based on the number of bytes. For LENGTH less than or equal to 4, Read to LINRD0 register will clear the “RXRDY” flag. For LENGTH greater than 4, Read to LINRD1 register will clear the “RXRDY” flag.

18.6.2 Transmitting Data

The SCI transmitter is enabled if 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.

18.6.2.1 Transmitting Data in Single-Buffer Mode

Single Buffer Mode is selected when MBUF MODE bit is 0. In this mode SCI waits for data to be written to SCITD, transfers it to SCITXSHF, and transmits it. The flags TXRDY and TX EMPTY 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.

User can transmit data by

1) Polling Transmit Ready Flag

2) Receive Interrupt

3) DMA

In polling method, software can poll for TXRDY bit to go high before writing the data to SCITD register. CPU is unnecessarily overloaded by doing this Polling method. To avoid this user can use either Interrupt or DMA method. To use interrupt method SET TX INT bit should be set and to use DMA SET TX DMA bit should be set. Either an Interrupt or a DMA request is generated the moment TXRDY 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 be done by either 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.

18.6.2.2 Transmitting Data in Multi-Buffer Mode

Multi-Buffer Mode is selected when MBUF MODE bit is 1. Similar to Single Buffer Mode the software can use polling, Interrupt or DMA method to write the data to be transmitted. The data to be transmitted has to be written to LINTD0 and LINTD1 register, based on the number of bytes. SCI waits for data to be written to Byte 0(TD0) of LINTD0 register and transfers the programmed number of bytes to SCITXSHF to transmit one by one automatically.

18.7 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.

18.8 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 18.14.

18.8.1 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

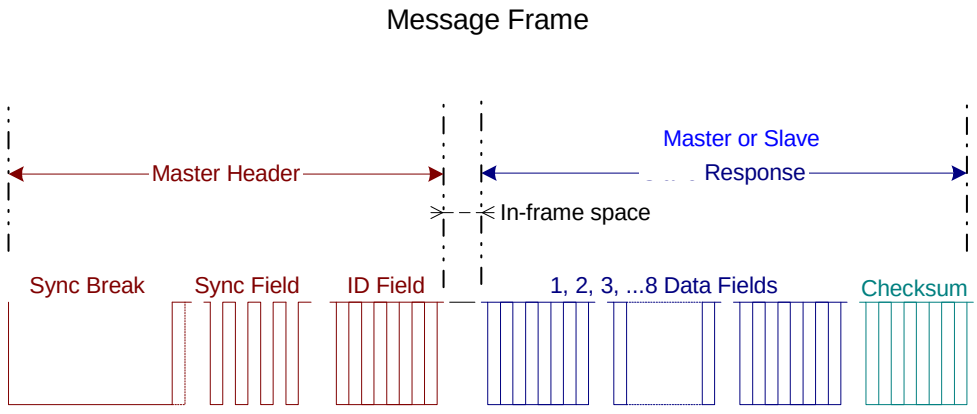
The LIN module covers the CPU performance-consuming features, defined in the *LIN Specification Package* Revision 1.3 and 2.0 by hardware.

The Master Mode of LIN module is compatible with LIN 2.1 standard.

18.8.2 Message Frame

The LIN protocol defines a message frame format, illustrated in Figure 18-12. 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.

Figure 18-12. LIN Protocol Message Frame Format: Master Header and Slave Response



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.

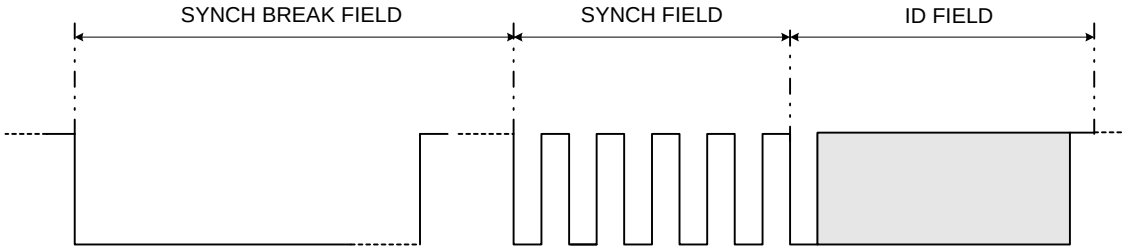
The LIN bus is a single channel wired-AND. The bus has a binary level: either dominant for a value of 0, or recessive for a value of 1.

18.8.2.1 Message Header

The header of a message is initiated by a master (see Figure 18-13) 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 18-13. Header 3 fields: Synch Break, Synch, and ID

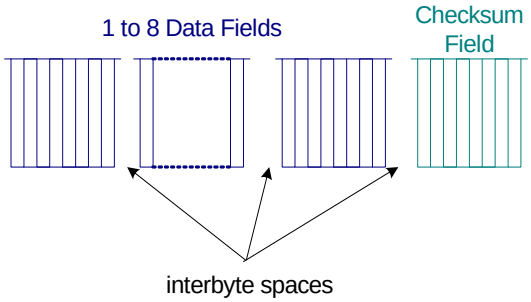


18.8.2.2 Response

The format of the response is as illustrated in Figure 18-14. 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 18-14. Response Format of LIN Message Frame

Response



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 18.8.6). 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 18-5, or by LENGTH value in SCIFORMAT[18:16] register; see Table 18-6. The SCI/LIN module supports response lengths from 1 to 8 bytes in compliance with LIN 2.0.

Table 18-5. 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 18-6. 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

18.8.3 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 BRSR 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.

18.8.4 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 (BRSR). There is an additional 3-bit fractional divider value, field U in the BRSR, which further fine-tunes the data field baud rate.

The ranges for the prescaler values in the BRSR register 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 BRSR register 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** T_{VCLK}" (with **a** = 0,1) is added to each T_{bit} as explained in Section 18.8.4.2. 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:

$$1\text{kHz} \le F_{\text{LINCLK}} \le 20\text{kHz}$$

All transmitted bits are shifted in and out at T_{bit} periods.

18.8.4.1 Fractional Divider

The M field of the BRSR register modifies the integer prescaler P for finer 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:

$$\text{For all P other than 0, and all M, } T_{bit} = 16\left(P + 1 + \frac{M}{16}\right)T_{VCLK}$$

$$\text{For P= 0 : } T_{bit} = 32T_{VCLK}$$

Therefore, the LINCLK frequency is given by

$$F_{LINCLK} = \frac{F_{VCLK}}{16\left(P + 1 + \frac{M}{16}\right)} \quad \text{For all P other than zero}$$

$$F_{LINCLK} = \frac{F_{VCLK}}{32} \quad \text{For P = 0}$$

18.8.4.2 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)

18.8.4.3 Superfractional Divider In LIN Mode

Building on the 4-bit fractional divider M (BRSR[27:24], the superfractional divider uses an additional 3-bit modulating value, illustrated in Table 18-7. 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 18-7. Superfractional Bit Modulation for LIN Master Mode and Slave Mode

BRS[30:28]	Start Bit	D[0]	D[1]	D[2]	D[3]	D[4]	D[5]	D[6]	D[7]	Stop Bit
0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	1	0
2	1	0	0	0	1	0	0	0	1	0
3	1	0	1	0	1	0	0	0	1	0
4	1	0	1	0	1	0	1	0	1	0
5	1	1	1	0	1	0	1	0	1	1
6	1	1	1	0	1	1	1	0	1	1
7	1	1	1	1	1	1	1	0	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 BRSR[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), $T^{i}_{bit} = \left[16\left(P + 1 + \frac{M}{16}\right) + d\right]T_{VCLK}$

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), $T^{a}_{bit} = \left[16\left(P + 1 + \frac{M}{16}\right) + d\right]T_{VCLK}$

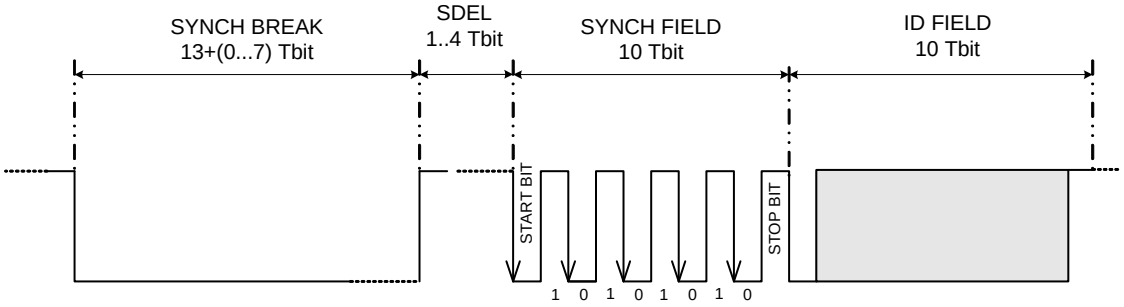
For P= 0 $T_{bit} = 32T_{VCLK}$

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.

18.8.5 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 18-15.

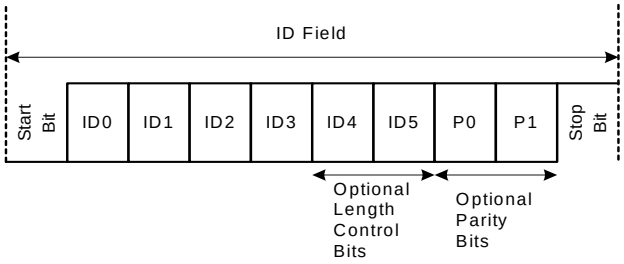
Figure 18-15. Message Header in Terms of T_{bit}



- 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 18-16 for an illustration of the ID field.

Figure 18-16. 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 multibuffer 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.

18.8.5.1 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.

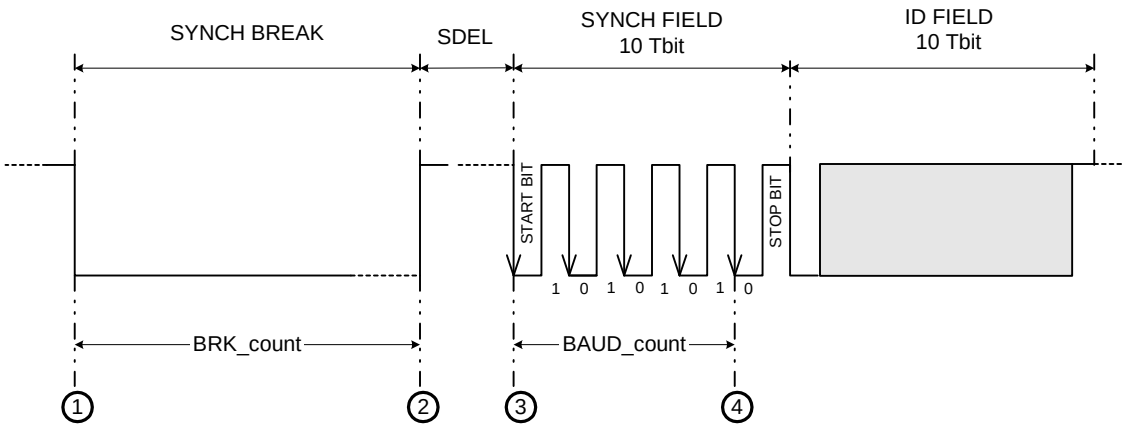
Even in the case of a collision, the received (corrupted) data is accessible in the RX buffers; registers LINRD0 and LINRD1.

18.8.5.2 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 18-17). These values are always calculated during the Header reception for synch field validation (Figure 18-18).

Figure 18-17. Measurements for Synchronization



By measuring the values BRK_count and BAUD_count, a valid synch break sequence can be detected as described in Figure 18-18. The four numbered events in Figure 18-17 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:

$$\text{BAUD_count} + \text{BAUD_count} \gg 2 + \text{BAUD_count} \gg 3 \leq \text{BRK_count}$$

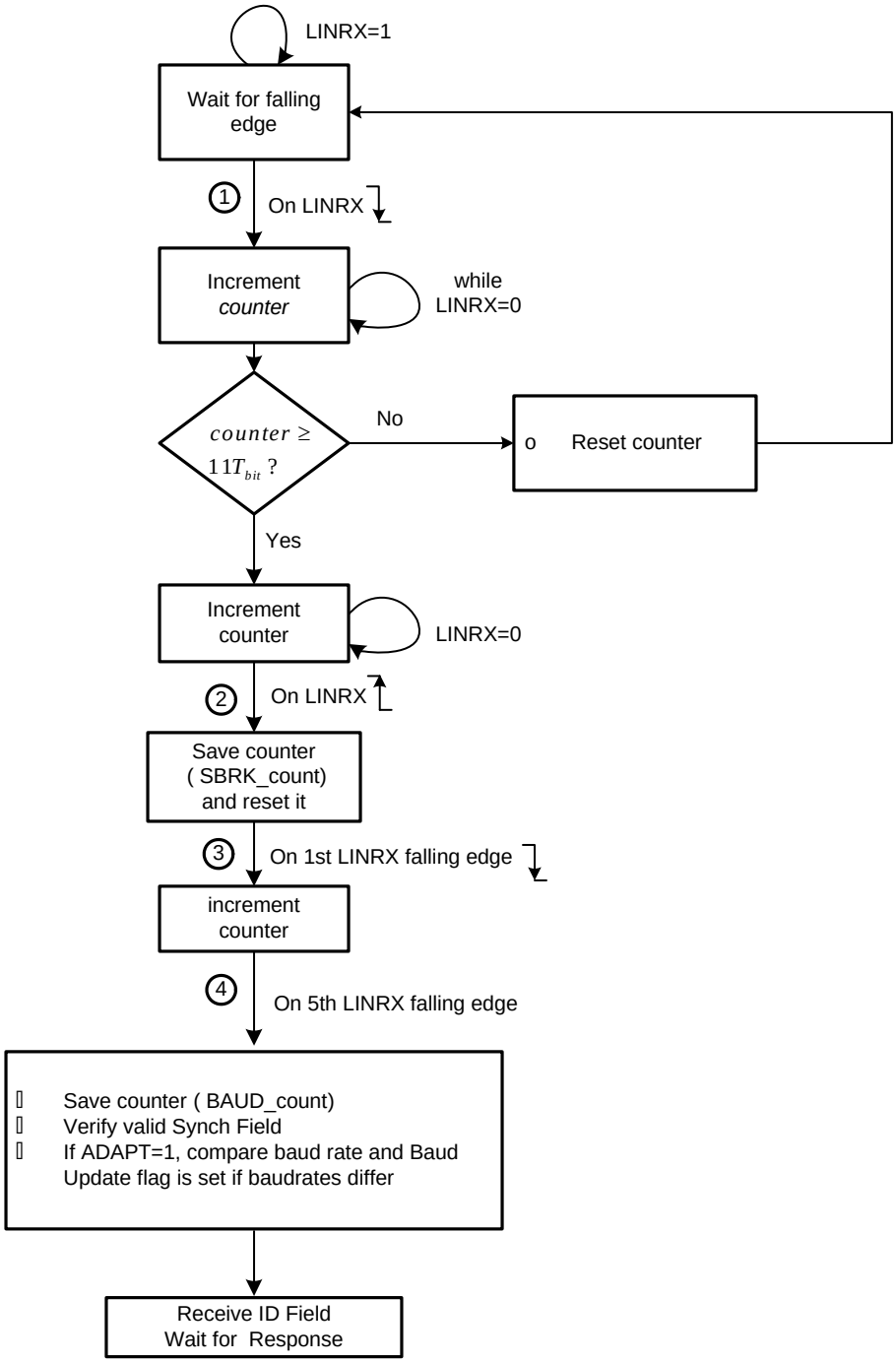
The BAUD_count value is shifted 3 times to the right and rounded using the first insignificant bit to obtain a T_{bit} unit. If the ADAPT bit is set, then the detected baud rate is compared to the programmed baud rate.

During the header reception processing as illustrated in Figure 18-18, 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 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 mistakingly be detected as a synch break.

Note:
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 18-18. Synchronization Validation Process and Baud Rate Adjustment



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.

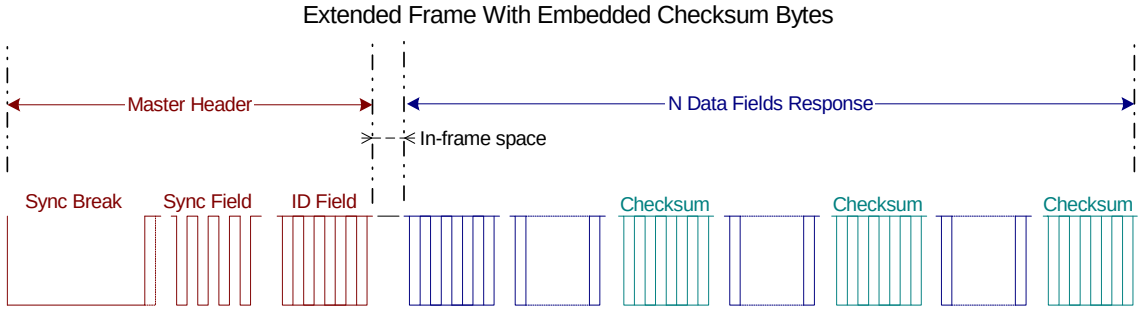
Note:
When an inconsistent synch field (ISFE) error occurs, suggested action for the application is to Reset the SWnRST bit and Set the SWnRST bit to make sure that the internal state machines are back to their normal states

18.8.6 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 18-19. 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 18-19. Optional Embedded Checksum in Response for Extended Frames



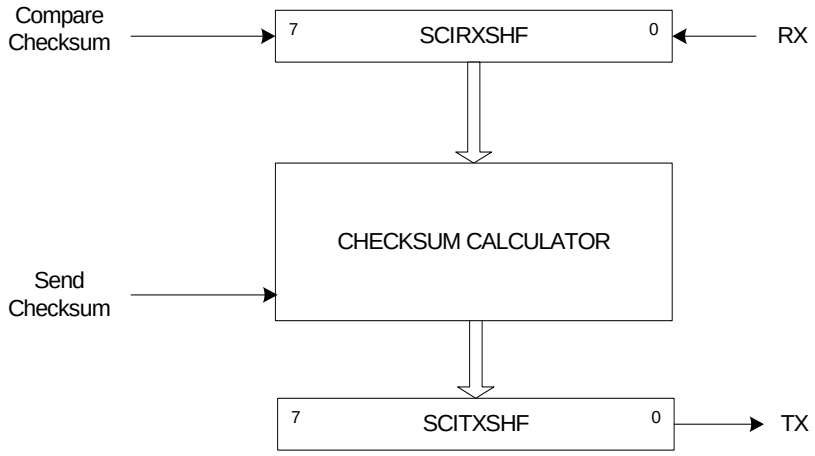
An ID interrupt will be generated (if enabled and there is a match) on reception of ID 62 (0x3E). This interrupt allows the CPU using a software counter to keep track of the bytes that are being sent out and decides when to calculate and insert a checksum byte (recommended at periodic rates). To handle 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 18-20.

Note:
The LIN 2.0 enhanced checksum does not apply to the reserved identifiers. The reserved identifiers always use the classic checksum.

Figure 18-20. Checksum Compare and Send for Extended Frames



18.8.7 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

18.8.7.1 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 18-8. 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

18.8.7.2 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:
If NRE error detection occurs before the bus idle detection (e.g. incomplete header is received and the bus is held recessive untill timeout), it can flag the bus idle timeout after 3.96 s (rather than a minimum of 4 s) at 1KHz baud rate (worst case).

18.8.7.3 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 18.12.3 for more details.

18.8.8 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:

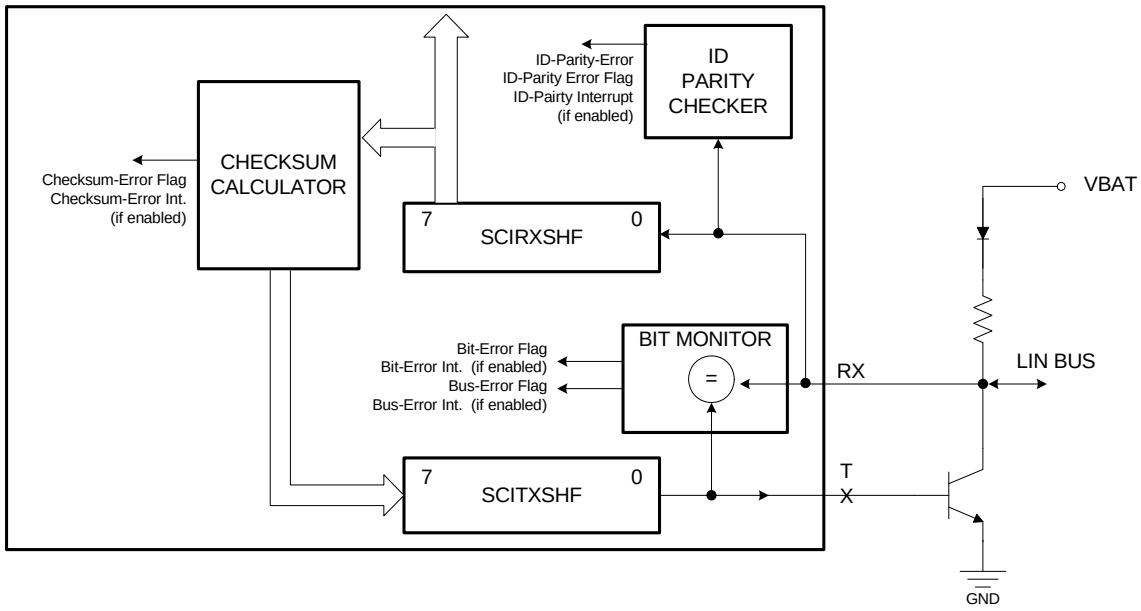
- Bit errors (BE)
- Physical bus errors (PBE)
- Identifier parity errors (PE)
- Checksum errors (CE)

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.

18.8.8.1 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 18-21.

Figure 18-21. TXRX Error Detector



18.8.8.2 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 (i.e. Bus shorted to GND or VBAT). The bit monitor detects a PBE during the header transmission, if no Synch Break can be generated (e.g. because of a bus shortage to VBAT) or if no Synch break Delimiter can be generated (e.g. 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.

18.8.8.3 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.

$$P0 = \overline{ID0 \oplus ID1 \oplus ID2 \oplus ID4} \text{ (even parity)}$$
$$P1 = \overline{ID1 \oplus ID3 \oplus ID4 \oplus ID5} \text{ (odd parity)}$$

If an ID-parity error is detected, the ID-parity error is flagged, and the received ID is not valid. See Section 18.8.9 for details.

18.8.8.4 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 18-22) for classic checksum implementation. The checksum byte is the inverted sum of the identifier byte and all the data bytes (see Figure 18-23) 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 18-22. Classic Checksum Generation at Transmitting Node

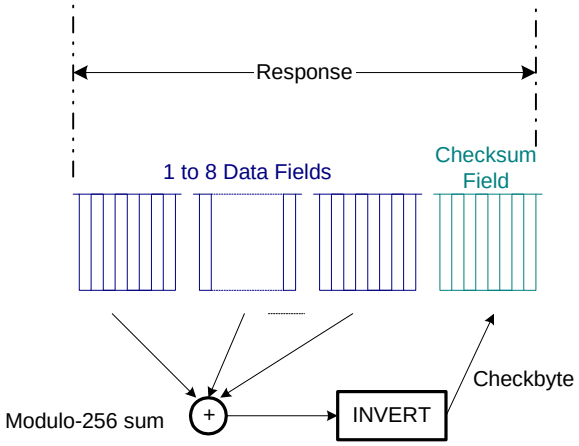
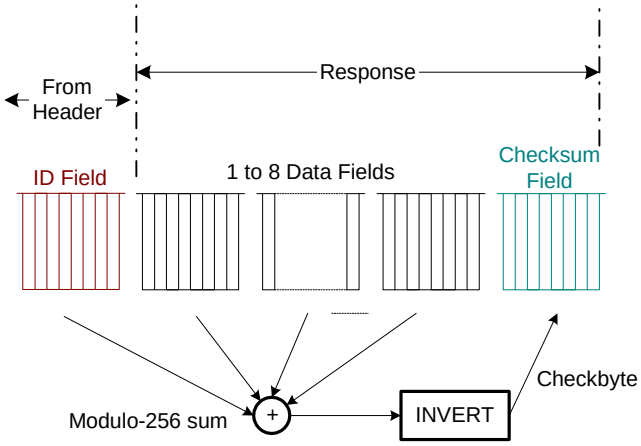


Figure 18-23. LIN 2.0-Compliant Checksum Generation at Transmitting Node



18.8.9 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 18-24. During header reception, all nodes filter the ID-Field (ID-Field is the part of the header explained in Figure 18-16) 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 18-24. 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; i.e., compare five most significant bits (MSBs) and filter three least significant bits (LSBs), the acceptance mask could be:

$$(0x26 + 0x25) \oplus 0x20 = 0x07$$

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.

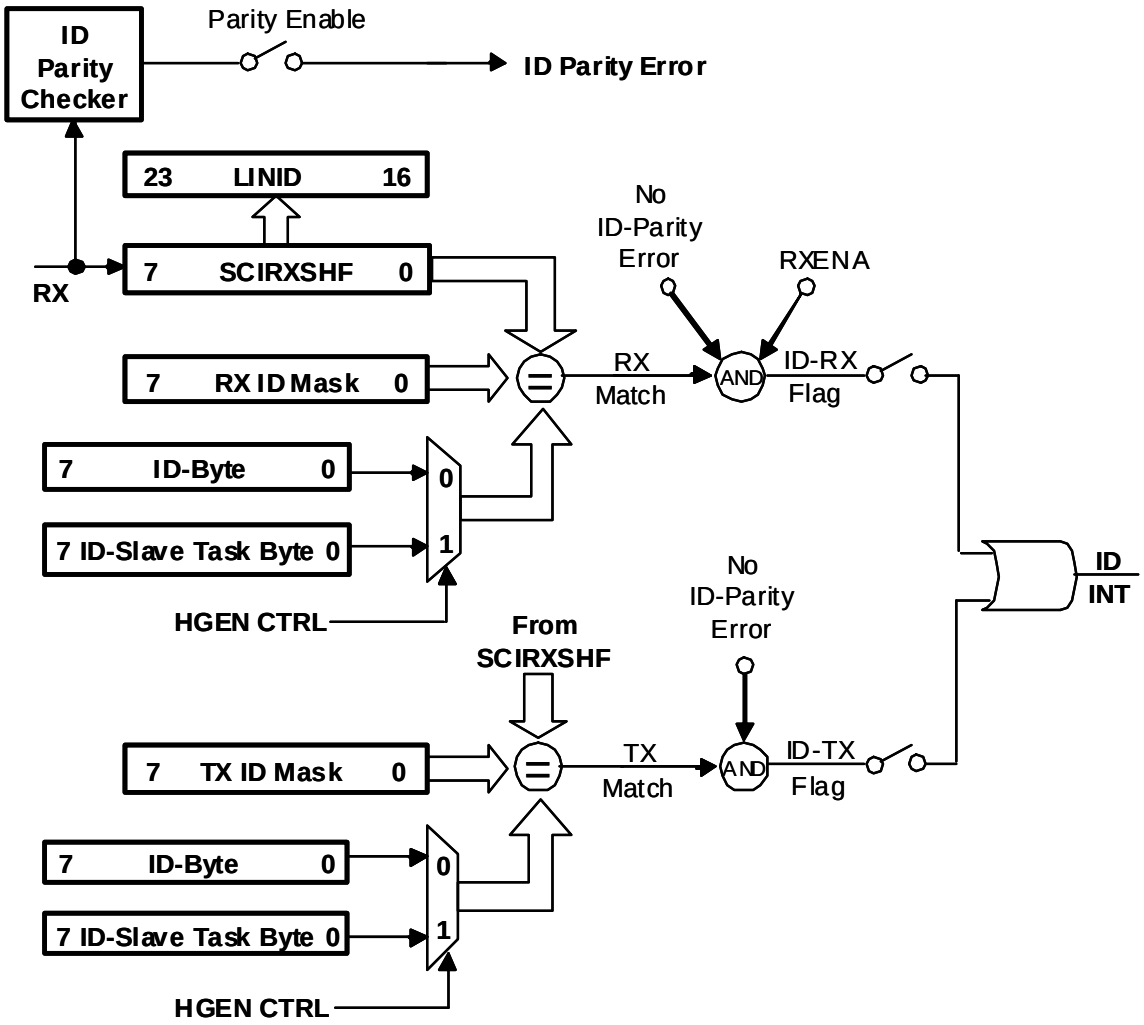
Note:
When byte 0 is written to TD0 (LINTD0[31:24]), the response transmission is automatically generated.

In multibuffer 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 multibuffer 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 multibuffer 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 multibuffer 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 18-24. ID Reception, Filtering and Validation



18.8.10 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 multibuffer 3-bit counter counts the data bytes transferred from the SCIRXSHF register to the RDy receive buffers if multibuffer mode is enabled, or to RD0 if multibuffer 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*), 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 18-8 illustrates the transmit 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 multibuffer mode is enabled or not(MBUF MODE bit).

18.8.11 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 multibuffer mode is not enabled (MBUF MODE bit).

The multibuffer 3-bit counter counts the data bytes transferred from the TDy transmit buffers register if multibuffer mode is enabled, or from TD0 to SCITXSHF if multibuffer 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*), 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 multibuffer mode is enabled or not (MBUF MODE bit).

Figure 18-9 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 multibuffer 3-bit counter counts the data bytes transferred from the TDy buffers into the SCITXSHF register.

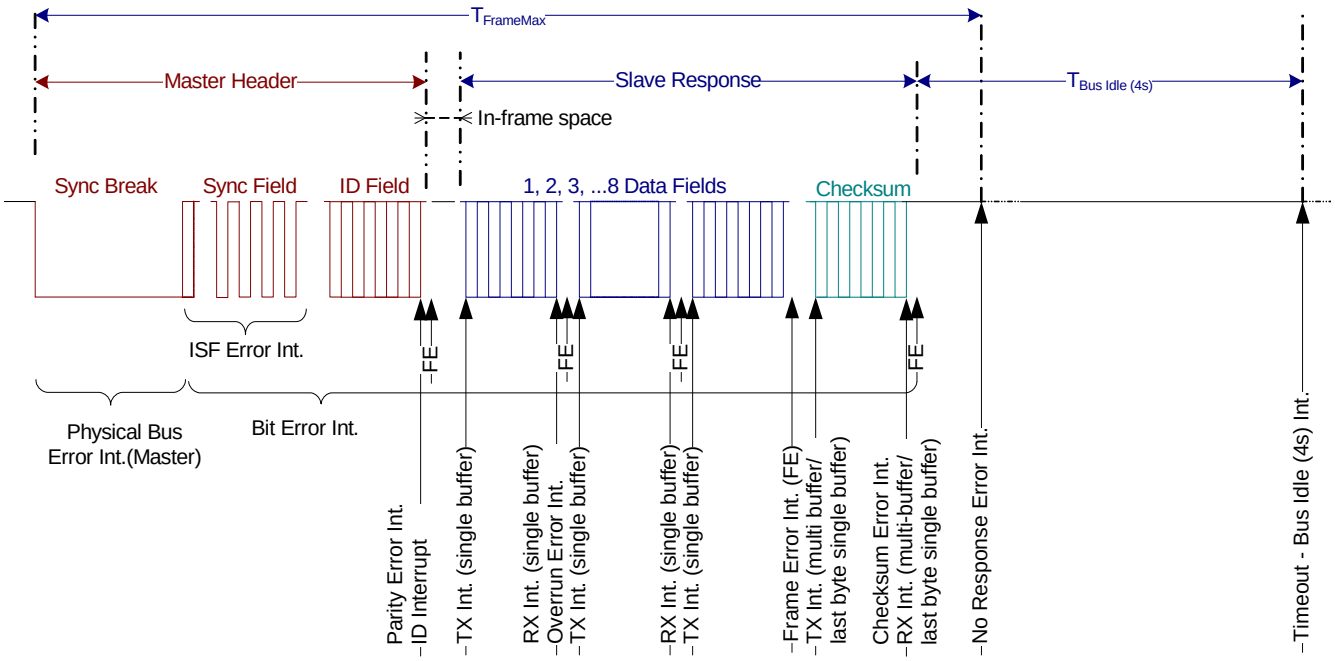
Note:
The transmit interrupt request can be eliminated until the next series of data is written into the transmit buffers LINTD0 and LINTD1, by disabling the corresponding interrupt via the SCICLRINT register or by disabling the transmitter via the TXENA bit.

18.9 LIN Interrupts

LIN and SCI mode have a common Interrupt block as explained in Section 18.4. There are 16 interrupt sources in the SCI/LIN module, with 8 of them being LIN mode only, as seen in Table 18-4.

A LIN message frame indicating the timing and sequence of the LIN interrupts that could occur is shown in Figure 18-25.

Figure 18-25. LIN Message Frame Showing LIN Interrupt Timing and Sequence



18.10 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 multibuffer mode is enabled or not via the multibuffer enable control bit (MBUF MODE).

18.10.1 LIN Receive DMA Requests

In LIN mode, when the multibuffer 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 multibuffer 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.

18.10.2 LIN Transmit DMA Requests

In LIN mode with the multibuffer 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 multibuffer 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.

18.11 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 SWnRST is held low the entire time that the LIN is being configured, the order in which the registers are programmed is not important

- Enable LIN by setting RESET bit.
- Clear SWnRST to 0 before configuring the LIN.
- Configure the LINRX and LINTX pins as SCI functional by setting the RX FUNC and TX FUNC bit.
- Select LIN mode by programming LIN MODE bit.
- Select Master or Slave mode by programming the CLOCK bit.
- Select the desired frame format(Checksum, Parity, length control) by programming SCIGCR1.
- Select multi-buffer mode by programming MBUF MODE bit.
- Select the baud rate to be used for communication by programming BRSR.
- Set the Maximum baud rate to be used for communication by programming BRSR.
- Set the CONT bit to make LIN not to 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 to connect the transmitter to the receiver internally. (This feature is used to perform a self-test.)
- Select the receiver enable RXENA bit if data is to be received.
- Select the transmit enable TXENA bit if data is to be transmitted.
- Select the RX ID MASK and the TX ID MASK fields in the LINMASK register.
- Set SWnRST to 1 after the SCI is configured.
- Perform Receive or Transmit data. (see Section 18.8.9 / Section 18.8.10 / Section 18.11.2)

18.11.1 Receiving Data

The LIN receiver is enabled to receive messages if 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/LIN function pin.

ID RX flag is set after a valid LIN ID is received with RX Match, generated ID interrupt if enabled.

18.11.1.1 Receiving Data in Single-Buffer Mode

Single Buffer Mode is selected when MBUF MODE bit is 0. In this mode SCI/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 FE, OE, or PE if any of these error conditions were detected in the received data. These error conditions are supported with configurable Interrupt capability.

User can read the Received data by

1) Polling Receive Ready Flag

2) Receive Interrupt

3) DMA

In polling method, software can poll for RXRDY bit and read the data from RD0 byte of LINRD0 register once RXRDY is set high. CPU is unnecessarly overloaded by selecting Polling mode. To avoid this user can use either Interrupt or DMA method. To use interrupt method SET RX INT bit should be set and to use DMA SET RX DMA bit should be set. Either an Interrupt or a DMA request is generated the moment RXRDY is set. If checksum scheme is used once Compare Checksum CC bit is set, the checksum will be compared on the byte that is currently being received, expected to be the checkbyte be enabled during the last byte of the data. CC bit will be cleared once Checksum is received. A CE will immediatly be flagged if there is a checksum error.

18.11.1.2 Receiving Data in Multi-Buffer Mode

Multi-Buffer Mode is selected when MBUF MODE bit is 1. In this mode SCI/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 same as Single Buffer mode, except that it monitors for the complete frame. Like Single Buffer mode the user can use either Interrupt, DMA or polling method to read the data. The received data has to be read from the LINRD0 and LINRD1 register, based on the number of bytes. For LENGTH less than or equal to 4, Read to LINRD0 register will clear the "RXRDY" flag. For LENGTH greater than 4, Read to LINRD1 register will clear the "RXRDY" flag. If checksum scheme is enabled by setting CC bit during the reception of the data, then the byte that is received after the reception of the programmed no. of data bytes indicated by LENGTH is treated as a checksum byte and compared.

18.11.2 Transmitting Data

The SCI transmitter is enabled if 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 TD0 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.

ID TX flag is set after a valid LIN ID is received with TX Match, generated ID interrupt if enabled.

18.11.2.1 Transmitting Data in Single-Buffer Mode

Single Buffer Mode is selected when MBUF MODE bit is 0. In this mode LIN waits for data to be written to TD0, transfers it to SCITXSHF, and transmits it. The flags TXRDY and TX EMPTY 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.

User can transmit data by

1) Polling Transmit Ready Flag

2) Receive Interrupt

3) DMA

In polling method, software can poll for TXRDY bit to go high before writing the data to TD0 register. CPU is unnecessarily overloaded by doing this Polling method. To avoid this user can use either Interrupt or DMA method. To use interrupt method SET TX INT bit should be set and to use DMA SET TX DMA bit should be set. Either an Interrupt or a DMA request is generated the moment TXRDY is set. When the SCI has completed transmission of all pending frames, the SCITXSHF register and 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 be done by either disabling the transmit interrupt(CLR TX INT) / DMA request (CLR TX DMA bit) or by disabling the transmitter (clear TXENA bit). In checksum scheme once Send Checksum SC bit is set, the checksum will be sent after the current transmission.

Note: The TXRDY flag cannot be cleared by reading the corresponding interrupt offset in the SCIINTVECT0 or SCIINTVECT1 register.

18.11.2.2 Transmitting Data in Multi-Buffer Mode

Multi-Buffer Mode is selected when MBUF MODE bit is 1. Similar to Single Buffer mode the software can use polling, Interrupt or DMA method to write the data to be transmitted. The data to be transmitted has to be written to LINTD0 and LINTD1 register, based on the number of bytes. SCI/LIN waits for data to be written to Byte 0(TD0) of LINTD0 register and transfers the programmed number of bytes to SCITXSHF to transmit one by one automatically. In checksum scheme once Send Checksum SC bit is set, the checksum will be sent after transmission of programmed no. of data bytes indicated by LENGTH field.

18.12 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.

The LIN module enters low-power mode when a sleep command frame is received. A wakeup signal will terminate the sleep mode of the LIN bus. On receipt of the sleep command, the POWERDOWN bit must be set by the application software to make the module enter local low-power mode.

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.

18.12.1 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.

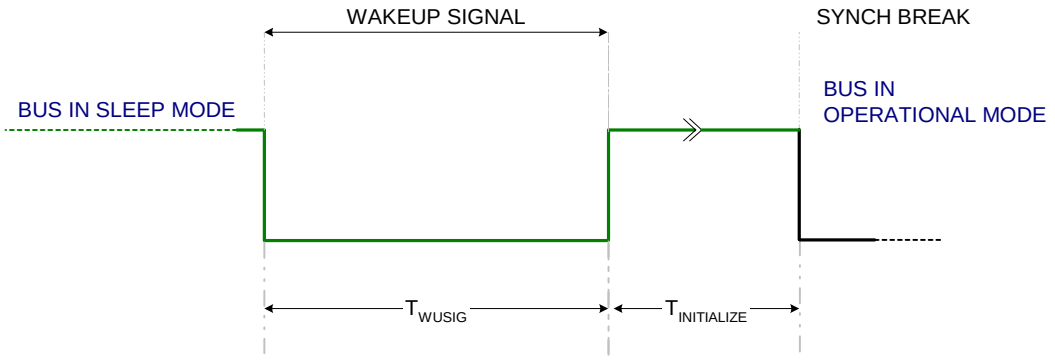
18.12.2 Wakeup

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 18-26. 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 18-26. Wakeup Signal Generation



$$0.25\text{ms} \leq T_{WUSIG} \leq 5\text{ms}$$

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 dominant level in 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).

18.12.3 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.

Note:
The LIN controller handles the wakeup expiration times defined by the LIN protocol with a hardware implementation.

18.13 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 sucessful. The suggested usage is to set CONT bit during emulation mode for sucessful communication.

18.14 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 Figure 18-27. 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.

Figure 18-27. SCI/LIN Control Registers Summary

Register Offset Address[1]	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
0x00 SCIGCR0 Page 971	Reserved															
	Reserved															RESET
0x04 SCIGCR1 Page 972	Reserved						TX ENA	RX ENA	Reserved						CONT	LOOP BACK
	Reserved		STOP EXT FRAME	HGEN CTRL	CTYPE	MBUF MODE	ADAPT	SLEEP	SW nRST	LIN MODE	CLOCK	STOP	PARITY	PAR-ITY ENA	TIM-MING MODE	COMM MODE
0x08 SCIGCR2 Page 979	Reserved								Reserved						CC	SC
	Reserved							GEN WU	Reserved							POWER DOWN
0x0C SCISETINT Page 981	SET BE INT	SET PBE INT	SET CE INT	SET ISFE INT	SET NRE INT	SET FE INT	SET OE INT	SET PE INT	Reserved					SET RX DMA ALL	SET RX DMA	SET TX DMA
	Reserved		SET ID	Reserved			SET RX INT	SET TX INT	SET TOA3 WUS INT	SET TOA WUS INT	Reserved	SET TIME-OUT INT	Reserved		SET WAKE UP INT	SET BRKDT INT
0x10 SCICLEARINT Page 986	CLR BE INT	CLR PBE INT	CLR CE INT	CLR ISFE INT	CLR NRE INT	CLR FE INT	CLR OE INT	CLR PE INT	Reserved					CLR RX DMA ALL	CLR RX DMA	CLR TX DMA
	Reserved		CLR ID INT	Reserved			CLR RX INT	CLR TX INT	CLR TOA3 WUS INT	CLR TOA WUS INT	Reserved	CLR TIME-OUT INT	Reserved		CLR WAKE UP INT	CLR BRKDT INT
0x14 SCI SETINTLVL Page 991	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	Reserved					SET RX DMA ALL INT LVL	Reserved	
	Reserved		SET ID INT LVL	Reserved			SET RX INT LVL	SET TX INT LVL	SET TOA3 WUS INT LVL	SET TOA WUS INT LVL	Reserved	SET TIME-OUT INT LVL	Reserved		SET WAKE UP INT LVL	SET BRKDT INT LVL

1 The offset address is relative to the peripheral's beginning address.

Register Offset Address[1]	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
0x18 SCICLEARINTLVL Page 995	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	Reserved					CLRRX DMA ALL INT LVL	Reserved	
	Reserved		CLR ID TX INT LVL	Reserved			CLR ID RX INT LVL	CLR TX INT LVL	CLR TOA3 WUS INT LVL	CLR TOA WUS INT LVL	Reserved	CLR TIME-OUT INT LVL	Reserved		CLR WAKE UP INT LVL	CLR BRKDT INT LVL
0x1C SCIFLR Page 999	BE	PBE	CE	ISFE	NRE	FE	OE	PE	Reserved							
	Reserved	ID RX	ID TX	RX WAKE	TX EMPTY	TX WAKE	RX RDY	TX RDY	TOA3 WUS	TOA WUS	Reserved	TIME-OUT	BUSY	IDLE	WAKE UP	BRKDT
0x20 SCIINTVECT0 Page 1011	Reserved															
	Reserved														INTVECT0[4:0]	
0x24 SCIINTVECT1 Page 1012	Reserved															
	Reserved														INTVECT1[4:0]	
0x28 SCIFORMAT Page 1013	Reserved															LENGTH[2:0]
	Reserved															CHAR[2:0]
0x2C BRS Page 1015	Reserved	U(2–0)			M(3–0)				PRESCALER P [23:16]							
	PRESCALER P [15:0]															
0x30 SCIED Page 1018	Reserved															
	Reserved								ED[7:0]							
0x34 SCIRD Page 1019	Reserved															
	Reserved								RD[7:0]							

1 The offset address is relative to the peripheral's beginning address.

Register Offset Address[1]	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
0x38 SCITD Page 1020	Reserved															
	Reserved								TD[7:0]							
0x3C SCIPIO0 Page 1021	Reserved															
	Reserved													TX FUNC	RX FUNC	CLK FUNC
0x40 SCIPIO1 Page 1022	Reserved															
	Reserved													TX DIR	RX DIR	CLK DIR
0x44 SCIPIO2 Page 1025	Reserved															
	Reserved													TX IN	RX IN	CLK IN
0x48 SCIPIO3 Page 1026	Reserved															
	Reserved													TX OUT	RX OUT	CLK OUT
0x4C SCIPIO4 Page 1028	Reserved															
	Reserved													TX SET	RX SET	CLK SET
0x50 SCIPIO5 Page 1030	Reserved															
	Reserved													TX CLR	RX CLR	CLK CLR
0x54 SCIPIO6 Page 1032	Reserved															
	Reserved													TX PDR	RX PDR	CLK PDR

1 The offset address is relative to the peripheral's beginning address.

Register Offset Address[1]	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
0x58 SCIPIO7 Page 1034	Reserved															
	Reserved													TX PD	RX PD	CLK PD
0x5C SCIPIO8 Page 1035	Reserved															
	Reserved													TX PSL	RX PSL	CLK PSL
0x60 LINCOMPARE Page 1036	Reserved															
	Reserved						SDEL[1:0]		Reserved					SBREAK[2:0]		
0x64 LINRD0 Page 1038	RD0[7:0]								RD1[7:0]							
	RD2[7:0]								RD3[7:0]							
0x68 LINRD1 Page 1039	RD4[7:0]								RD5[7:0]							
	RD6[7:0]								RD7[7:0]							
0x6C LINMASK Page 1040	Reserved								RX ID MASK[7:0]							
	Reserved								TX ID MASK[7:0]							
0x70 LINID Page 1041	Reserved								Received ID[7:0]							
	ID-SlaveTask BYTE(7–0)								ID BYTE[7:0]							
0x74 LINTD0 Page 1042	TD0[7:0]								TD1[7:0]							
	TD2[7:0]								TD3[7:0]							

1 The offset address is relative to the peripheral's beginning address.

Register Offset Address[1]	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
0x78 LINTD1 Page 1043	TD4[7:0]								TD5[7:0]							
	TD6[7:0]								TD7[7:0]							
0x7C MBRS Page 1044	Reserved															
	Reserved				MBR[12:0]											
0x80 Reserved	Reserved															
	Reserved															
0x84 Reserved	Reserved															
	Reserved															
0x88 Reserved	Reserved															
	Reserved															
0x8C Reserved	Reserved															
	Reserved															
0x90 IODFTCTRL Page 1045	BEN	PBEN	CEN	ISFE	Reserved	FEN	PEN	BRKDT ENA	Reserved			PIN SAMPLE MASK		TX SHIFT[2:0]		
	Reserved				IODFTENA[3:0]				Reserved						LPB ENA	RXP ENA

1 The offset address is relative to the peripheral's beginning address.

18.14.1 SCI Global Control Register 0 (SCIGCR0)

The SCIGCR0 register defines the module reset. Figure 18-28 and Table 18-9 illustrate this register.

Figure 18-28. SCI Global Control Register 0 (SCIGCR0) [offset = 0x00]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															RESET
R-0															RWP-0

R = Read, W = Write in all modes; RWP = Read/Write in privileged mode only; S = Set, U = Undefined, *-n* = Value after reset

Table 18-9. SCI Global Control Register 0 (SCIGCR0) Field Description

Bit	Name	Value	Description
31–1	Reserved		Reads return 0 and 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.

18.14.2 SCI Global Control Register 1 (SCIGCR1)

The SCIGCR1 register defines the frame format, protocol, and communication mode used by the SCI. Figure 18-29 and Table 18-10 illustrate this register.

Figure 18-29. SCI Global Control Register 1 (SCIGCR1) [offset = 0x04]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						TXENA	RXENA	Reserved						CONT	LOOP BACK
R-0						R/W-0	R/W-0	R-0						R/W-0	R/W-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		STOP EXT F RAME	HGEN CTRL	CTYPE	MBUF MODE	ADAPT	SLEEP	SW nRST	LIN MODE	CLOCK	STOP	PARITY	PARITY ENA	TIMING MODE	COMM MODE
R-0	R-0	R/WL-0	R/WL-0	R/WL-0	R/W-0	R/WL-0	R/W-0	R/W-0	RW-0	R/W-0	R/WC-0	R/WC-0	R/W-0	R/WC-0	R/W-0

R = Read in all modes; W = Write in all modes; WL = Write in LIN mode only; WP = Write in privileged mode only; WC = Write in sci-compatible mode only; *-n* = Value after reset

Table 18-10. SCI Global Control Register 1 (SCIGCR1) Field Description

Bit	Name	Value	Description
31–26	Reserved		Reads return 0 and 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).

Table 18-10. SCI Global Control Register 1 (SCIGCR1) Field Description (Continued)

Bit	Name	Value	Description
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 multibuffers.
		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 18-9) 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		Reads return 0 and writes have no effect.
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.

Table 18-10. SCI Global Control Register 1 (SCIGCR1) Field Description (Continued)

Bit	Name	Value	Description
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		Reads return 0 and 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 LINID occurs. Mask of 0xFF in LINMASK register will result in no match.
		1	ID filtering uses ID-SlaveTask BYTE (recommended). Mask of 0xFF in 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 0xFF and the TX ID MASK must be set to 0xFF.**
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		Multibuffer mode. This bit is effective in LIN and SCI modes.This bit controls receive/transmit buffer usage, i.e., whether the RX/TX multibuffers are used or a single register, RD0/TD0, is used.
		0	The multi-buffer mode is disabled.
		1	The multi-buffer mode is enabled.

Table 18-10. SCI Global Control Register 1 (SCIGCR1) Field Description (Continued)

Bit	Name	Value	Description
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.
8	SLEEP		SCI sleep. This bit is effective in LIN and SCI modes. 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 18-9) 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 18.12 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 18-11 and Table 18-12. 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 SW nRESET = 0.

Table 18-10. SCI Global Control Register 1 (SCIGCR1) Field Description (Continued)

Bit	Name	Value	Description
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.

Table 18-10. SCI Global Control Register 1 (SCIGCR1) Field Description (Continued)

Bit	Name	Value	Description
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.
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.

Table 18-10. SCI Global Control Register 1 (SCIGCR1) Field Description (Continued)

Bit	Name	Value	Description
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 18-11. SCI Receiver Status Flags

SCI Flag	Register	Bit	Value After SW nRESET(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 SW nRESET = 0.

Table 18-12. SCI Transmitter Status Flags

SCI Flag	Register	Bit	Value After SW nRESET(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 SW nRESET = 0.

18.14.3 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 18-30 and Table 18-13 illustrate this register.

Figure 18-30. SCI Global Control Register 2 (SCIGCR2) [offset = 0x08

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved														CC	SC
R-0														R/WL-0	R/WL-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							GEN WU	Reserved							POWER DOWN
R-0							R/W-0	R-0							R/W-0

R = Read, W = Write in all modes, WL = Write in LIN mode only; *-n* = Value after reset

Table 18-13. SCI Global Control Register 2 (SCIGCR2) Field Description

Bit	Name	Value	Description
31–18	Reserved		Reads return 0 and 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. In non-multibuffer 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 (i.e., during the inter-frame space), then the immediate next byte will be treated as a checksum byte. A CE will immediately be flagged if there is a checksum error. This bit is automatically cleared once the checksum is compared. See Section 18.8.6 for more details.
		0	No checksum compare will occur.
		1	Compare checksum on expected checksum byte.

Table 18-13. SCI Global Control Register 2 (SCIGCR2) Field Description (Continued)

Bit	Name	Value	Description
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-multibuffer mode, the checksum byte will be sent after the current byte transmission. In multibuffer mode, the checksum byte will be sent after the last byte count, indicated by the SCIFORMAT[18:16]). See Section 18.8.6 for more details. This byte will be cleared after the checksum byte has been transmitted.
		0	No checksum byte will be sent.
		1	A checksum byte will be sent.
15–9	Reserved		Reads return 0 and 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		Reads return 0 and writes have no effect.
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 18.12 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.

18.14.4 SCI Set Interrupt Register (SCISETINT)

Figure 18-31 and Table 18-14 illustrate this register. Refer Figure 18-31 for details on when different interrupt flags get set in a frame during LIN Mode.

Figure 18-31. SCI Set Interrupt Register (SCISETINT) [offset = 0x0C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SET BE INT	SET PBE INT	SET CE INT	SET ISFE INT	SET NRE INT	SET FE INT	SET OE INT	SET PE INT	Reserved					SET RX DMA ALL	SET RX DMA	SET TX DMA
R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/W-0	R/W-0	R/W-0	R-0					R/WC-0	R/W-0	R/W-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		SET ID INT	Reserved			SET RX INT	SET TX INT	SET TOA3 WUS INT	SET TOA WUS INT	Reserved	SET TIME-OUT INT	Reserved		SET WAKE UP INT	SET BRKDT INT
R-0		R/WL-0	R-0			R/W-0	R/W-0	R/WL-0	R/WL-0	R-0	R/WL-0	R-0		R/W-0	R/WC-0

R = Read in all modes; W = Write in all modes; WL = Write in LIN mode only; WC = Write in sci-compatible mode only; -*n* = Value after reset

Table 18-14. SCI Set Interrupt Register (SCISETINT) Field Description

Bit	Name	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.

Table 18-14. SCI Set Interrupt Register (SCISETINT) Field Description (Continued)

Bit	Name	Value	Description
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.
26	SET FE INT		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.
		0	*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		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.
		0	*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		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.
		0	*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		Reads return 0 and writes have no effect.

Table 18-14. SCI Set Interrupt Register (SCISETINT) Field Description (Continued)

Bit	Name	Value	Description
18	SET RX DMA ALL		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.
		0	*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		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 bit SCISETINT
		0	*Read:* Receive DMA request is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* Receive DMA request is enabled.
16	SET TX DMA		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 SET TX INT bit (SCISETINT).
		0	*Read:* Transmit DMA request is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* Transmit DMA request is enabled.
15–14	Reserved		Reads return 0 and writes have no effect.
13	SET ID INT		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 18.8.9 for more details.
		0	*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		Reads return 0 and writes have no effect.

Table 18-14. SCI Set Interrupt Register (SCISETINT) Field Description (Continued)

Bit	Name	Value	Description
9	SET RX INT		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.
		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. 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		Reads return 0 and 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		Reads return 0 and writes have no effect.

Table 18-14. SCI Set Interrupt Register (SCISETINT) Field Description (Continued)

Bit	Name	Value	Description
1	SET WAKEUP INT		Set wakeup interrupt. This bit is effective in LIN or SCI-compatible mode. Setting this bit enables the SCI/LIN to generate a wake-up interrupt and thereby exit low-power mode. If enabled, the wake-up 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.

18.14.5 SCI Clear Interrupt Register (SCICLEARINT)

Figure 18-32 and Table 18-15 illustrate this register. //ML: Should we explain purpose of set/clear registers here?

Figure 18-32. SCI Clear Interrupt Register (SCICLEARINT) [offset = 0x10]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CLR BE INT	SET PBE INT	CLR CE INT	CLR ISFE INT	CLR RE INT	CLR FE INT	CLR OE INT	CLR PE INT	Reserved					CLR RX DMA ALL	CLR RX DMA	CLR TX DMA
R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/W-0	R/W-0	R/W-0	R-0					R/WC-0	R/W-0	R/W-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		CLR ID INT	Reserved			CLR RX INT	CLR TX INT	CLR TOA3 WUS INT	CLR TOA WUS INT	Reserved	CLR TIME-OUT INT	Reserved		CLR WAKE UP INT	CLR BRKDT INT
R-0		R/WL-0	R-0			R/W-0	R/W-0	R/WL-0	R/WL-0	R-0	R/WL-0	R-0		R/W-0	R/WC-0

R = Read in all modes; W = Write in all modes; WC = Write in sci-compatible mode only; WL = Write in LIN mode only; -*n* = Value after reset

Table 18-15. SCI Clear Interrupt Register (SCICLEARINT) Field Description

Bit	Name	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		Set 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.

Table 18-15. SCI Clear Interrupt Register (SCICLEARINT) Field Description (Continued)

Bit	Name	Value	Description
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.
27	CLR NRE INT		Clear no-response-error interrupt. This bit is effective in LIN mode only. This bit disables the NRE 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.
26	CLR FE INT		Clear framing-error interrupt. This bit is effective in LIN or SCI mode. 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 OE INT		Clear overrun-error interrupt. This bit is effective in LIN or SCI mode. This bit disables the SCI/LIN 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 is effective in LIN or SCI mode. 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		Reads return 0 and writes have no effect.

Table 18-15. SCI Clear Interrupt Register (SCICLEARINT) Field Description (Continued)

Bit	Name	Value	Description
18	CLR RX DMA ALL		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.
		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.
17	CLR RX DMA		Clear receive DMA request. This bit is effective in LIN or SCI mode. This bit disables the receive DMA request when set.
		0	*Read:* The receive 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 is disabled.
16	CLR TX DMA		Clear transmit DMA request. This bit is effective in LIN or SCI mode. This bit disables the transmit DMA request when set.
		0	*Read:* The 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 is disabled.
15–14	Reserved		Reads return 0 and writes have no effect.
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		Reads return 0 and 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.

Table 18-15. SCI Clear Interrupt Register (SCICLEARINT) Field Description (Continued)

Bit	Name	Value	Description
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		Reads return 0 and 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		Reads return 0 and 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 18-15. SCI Clear Interrupt Register (SCICLEARINT) Field Description (Continued)

Bit	Name	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.

18.14.6 SCI Set Interrupt Level Register (SCISETINTLVL)

Figure 18-33 and Table 18-16 illustrate this register.

Figure 18-33. SCI Set Interrupt Level Register (SCISETINTLVL) [offset = 0x14

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
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	Reserved					SET RX DMA ALL INT LVL	Reserved	
R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/W-0	R/W-0	R/W-0	R-0					R/WC-0	R-0	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		SET ID INT LVL	Reserved			SET RX INT LVL	SET TX INT LVL	SET TOA3 WUS INT LVL	SET TOA WUS INT LVL	Reserved	SET TIME-OUT INT LVL	Reserved		SET WAKE UP INT LVL	SET BRKDT INT LVL
R-0		R/WL-0	R-0			R/W-0	R/W-0	R/WL-0	R/WL-0	R-0	R/WL-0	R-0		R/W-0	R/WC-0

R = Read in all modes; W = Write in all modes; WC = Write in sci-compatible mode only; WL = Write in LIN mode only; -*n* = Value after reset

Table 18-16. SCI Set Interrupt Level Register (SCISETINTLVL) Field Description

Bit	Name	Value	Description
31	SET BE INT LV		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.

Table 18-16. SCI Set Interrupt Level Register (SCISETINTLVL) Field Description (Continued)

Bit	Name	Value	Description
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.
25	SET OE INT LVL		Set overrun-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.
24	SET PE INT LVL		Set parity 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.
23–19	Reserved		Reads return 0 and writes have no effect.
18	SET RX DMA ALL INT LVL		Set receive DMA all interrupt levels. This bit is effective in SCI mode only.
		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 or write:* The receive interrupt request for address frames is mapped to the INT1 line.
17–14	Reserved		Reads return 0 and writes have no effect.
13	SET ID INT LVL		Set ID 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.
12–10	Reserved		Reads return 0 and writes have no effect.

Table 18-16. SCI Set Interrupt Level Register (SCISETINTLVL) Field Description (Continued)

Bit	Name	Value	Description
9	SET RX INT LVL		Set receiver 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.
8	SET TX INT LVL		Set transmitter 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.
7	SET TOA3WUS INT LVL		Set timeout after three wakeup signals 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.
6	SET TOAWUS INT LVL		Set timeout after wakeup signal 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.
5	Reserved		Reads return 0 and writes have no effect.
4	SET TIMEOUT INT LVL		Set timeout 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.
3–2	Reserved		Reads return 0 and writes have no effect.
1	SET WAKEUP INT LVL		Set wake-up 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 18-16. SCI Set Interrupt Level Register (SCISETINTLVL) Field Description (Continued)

Bit	Name	Value	Description
0	SET BRKDT INT		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.

18.14.7 SCI Clear Interrupt Level Register (SCICLEARINTLVL)

Figure 18-34 and Table 18-17 illustrate this register.

Figure 18-34. SCI Clear Interrupt Level Register (SCICLEARINTLVL) [offset = 0x18

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
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	Reserved					CLRRX DMA ALL INT LVL	Reserved	
R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/W-0	R/W-0	R/W-0	R-0					R/WC-0	R-0	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		CLR ID IX INT LVL	Reserved			CLR ID RX INT LVL	CLR TX INT LVL	CLR TOA3 WUS INT LVL	CLR TOA WUS INT LVL	Reserved	CLR TIME-OUT INT LVL	Reserved		CLR WAKE UP INT LVL	CLR BRKDT INT LVL
R-0		R/WL-0	R-0			R/W-0	R/W-0	R/WL-0	R/WL-0	R-0	R/WL-0	R-0		R/W-0	R/WC-0

R = Read in all modes; W = Write in all modes; WC = Write in sci-compatible mode only; WL = Write in LIN mode only; -*n* = Value after reset

Table 18-17. SCI Clear Interrupt Level Register (SCICLEARINTLVL) Field Description

Bit	Name	Value	Description
31	CLR BE INT LVL		Clear 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:* 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 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:* 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 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:* The interrupt level is mapped to the INT1 line. *Write:* The interrupt level is mapped to the INT0 line.

Table 18-17. SCI Clear Interrupt Level Register (SCICLEARINTLVL) Field Description (Continued)

Bit	Name	Value	Description
27	CLR NRE INT LVL		Clear 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:* The interrupt level is mapped to the INT1 line. *Write:* The interrupt level is mapped to the INT0 line.
26	CLR FE INT LVL		Clear 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:* The interrupt level is mapped to the INT1 line. *Write:* The interrupt level is mapped to the INT0 line.
25	CLR OE INT LVL		Clear overrun-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:* 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 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:* The interrupt level is mapped to the INT1 line. *Write:* The interrupt level is mapped to the INT0 line.
23–19	Reserved		Reads return 0 and writes have no effect.
18	CLR RX DMA ALL INT LVL		Clear receive DMA interrupt level. This bit is effective in SCI-compatible mode only.
		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.
17–14	Reserved		Reads return 0 and writes have no effect.

Table 18-17. SCI Clear Interrupt Level Register (SCICLEARINTLVL) Field Description (Continued)

Bit	Name	Value	Description
13	CLR ID INT LVL		Clear ID 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:* The interrupt level is mapped to the INT1 line. *Write:* The interrupt level is mapped to the INT0 line.
12–10	Reserved		Reads return 0 and writes have no effect.
9	CLR RX INT LVL		Clear receiver 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:* 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 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:* The interrupt level is mapped to the INT1 line. *Write:* The interrupt level is mapped to the INT0 line.
7	CLR TOA3WUS INT LVL		Clear timeout after three wakeup signals 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:* The interrupt level is mapped to the INT1 line. *Write:* The interrupt level is mapped to the INT0 line.
6	CLR TOAWUS INT LVL		Clear timeout after wakeup signal 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:* The interrupt level is mapped to the INT1 line. *Write:* The interrupt level is mapped to the INT0 line.
5	Reserved		Reads return 0 and writes have no effect.
4	CLR TIMEOUT INT LVL		Clear timeout 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:* The interrupt level is mapped to the INT1 line. *Write:* The interrupt level is mapped to the INT0 line.

Table 18-17. SCI Clear Interrupt Level Register (SCICLEARINTLVL) Field Description (Continued)

Bit	Name	Value	Description
3–2	Reserved		Reads return 0 and writes have no effect.
1	CLR WAKEUP INT LVL		Clear wake-up 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:* The interrupt level is mapped to the INT1 line. *Write:* The interrupt level is mapped to the INT0 line.
0	CLR BRKDT INT		Clear 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:* The interrupt level is mapped to the INT1 line. *Write:* The interrupt level is mapped to the INT0 line.

18.14.8 SCI Flags Register (SCIFLR)

Figure 18-35 and Table 18-18 illustrate this register.

Figure 18-35. SCI Flags Register (SCIFLR) [offset = 0x1C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BE	PBE	CE	ISFE	NRE	FE	OE	PE	Reserved							
R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/W-0	R/W-0	R/W-0	R-0							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	ID RX	ID TX	RX WAKE	TX EMPTY	TX WAKE	RX RDY	TX RDY	TOA3 WUS	TOA WUS	Reserved	TIME-OUT	BUSY	IDLE	WAKE UP	BRKDT
R-0	R/WL-0	R/WL-0	R/WC-0	R/W-1	R/WC-0	R/W-0	R/W-1	R/WL-0	R/WL-0	R-0	R/WL-0	R/W-0	R-0	R/WL-0	R/WC-0

R = Read in all modes; W = Write in all modes; WC = Write in sci-compatible mode only; WL = Write in LIN mode only; -*n* = Value after reset

Table 18-18. SCI Flags Register (SCIFLR) Field Description

Bit	Name	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 18.8.8 for more information. The bit error flag is cleared by any of the following: • Setting of the SW nRST 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.

Table 18-18. SCI Flags Register (SCIFLR) Field Description (Continued)

Bit	Name	Value	Description
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 18.8.8 for more information. The physical bus 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 • 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 (e.g. because of a bus shortage to VBAT) or if no Synch Break Delimiter can be generated (e.g. 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.
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 18.8.8 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 SW nRST 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.

Table 18-18. SCI Flags Register (SCIFLR) Field Description (Continued)

Bit	Name	Value	Description
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 18.8.5.2 for more information.The inconsistent synch field 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 • 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:* This 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 18.8.7 for more information. The no-response 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 • 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 18-18. SCI Flags Register (SCIFLR) Field Description (Continued)

Bit	Name	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 (Section 18.14.4). 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/frame, depending on whether the module is in SCI compatible or LIN mode In multibuffer 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. *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 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. *Write:* The bit is cleared to 0.

Table 18-18. SCI Flags Register (SCIFLR) Field Description (Continued)

Bit	Name	Value	Description
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 SW nRST 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		Reads return 0 and writes have no effect.
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 18.8.9 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 SW nRST 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:* This bit is cleared to 0.

Table 18-18. SCI Flags Register (SCIFLR) Field Description (Continued)

Bit	Name	Value	Description
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 18.8.9 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 SW nRST 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 a TX match. *Write:* This 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 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.

Table 18-18. SCI Flags Register (SCIFLR) Field Description (Continued)

Bit	Name	Value	Description
11	TX EMPTY		Transmitter empty flag. This flag indicates the transmitter's buffer register(s) (SCITD/TDy) and shift register (SCITXSHF) are empty. In multibuffer mode, this flag indicates the TDx registers and shift register (SCITXSHF) are empty. In non-multibuffer mode, this flag indicates the LINTD0 byte and the 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.** *SCI mode or LIN non-multibuffer 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 multibuffer mode:
		0	Multibuffer or shift register (or all) are loaded with data
		1	Multibuffer and shift registers are all empty.
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 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 18-18. SCI Flags Register (SCIFLR) Field Description (Continued)

Bit	Name	Value	Description
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 multibuffer mode, a valid frame being a message frame received with no errors. In *non-multibuffer 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 interrupt-enable bit is set (SCISETINT[9]); 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 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.

Table 18-18. SCI Flags Register (SCIFLR) Field Description (Continued)

Bit	Name	Value	Description
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 multibuffer 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 interrupt enable bit TXINT is set. **Note:** **1) TXRDY is also set to 1 either by setting of the RESET bit, enabling SW nRST 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	The multibuffers are full.
		1	The multibuffers are ready to receive the next character(s).
			For more information on transmit interrupt handling, see the SCI document for compatibility mode and Section 18.8.9 for LIN mode.

Table 18-18. SCI Flags Register (SCIFLR) Field Description (Continued)

Bit	Name	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 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 See Section 18.12.3 for more information.
		0	*Read:* No timeout occured after three wakeup signals. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* Timeout occured after 3 wakeup signals and 1.5 seconds time. *Write:* The bit will be 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 SW nRST 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 18.12.3 for more information.
		0	*Read:* No timeout occurs after one wakeup signal (150 ms). *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* A timeout occur after one wakeup signal. *Write:* The bit will be cleared to 0.
5	Reserved		Reads return 0 and 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 4 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 SW nRST 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 18.8.7 for more information.
		0	*Read:* No bus idle has been detected since this bit was cleared. *Write:* A write of 0 to this bit has no effect.
		1	*Read:* A LIN bus idle has been detected. *Write:* The bit is cleared to 0.

Table 18-18. SCI Flags Register (SCIFLR) Field Description (Continued)

Bit	Name	Value	Description
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 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. 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		Wake-up 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 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.

Table 18-18. SCI Flags Register (SCIFLR) Field Description (Continued)

Bit	Name	Value	Description
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 BRKDT INT ENA bit 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:* Writing a 1 to this bit clears it to 0.

18.14.9 SCI Interrupt Vector Offset 0 (SCIINTVECT0)

Figure 18-36 and Table 18-19 illustrate this register.

Figure 18-36. SCI Interrupt Vector Offset 0 (SCIINTVECT0) [offset = 0x20]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											INTVECT0[4:0]				
R-0											R-0				

R = Read in all modes; *-n* = Value after reset

Table 18-19. SCI Interrupt Vector Offset 0 (SCIINTVECT0) Field Description

Bit	Name	Value	Description
31–5	Reserved		Reads return 0 and writes have no effect.
4–0	INTVECT0[4:0]	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 18-4 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).**

18.14.10SCI Interrupt Vector Offset 1 (SCIINTVECT1)

Figure 18-37 and Table 18-20 illustrate this register.

Figure 18-37. SCI Interrupt Vector Offset 1 (SCIINTVECT1) [offset = 0x24

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											INTVECT1[4:0]				
R-0											R-0				

R = Read in all modes; -*n* = Value after reset

Table 18-20. SCI Interrupt Vector Offset 1 (SCIINTVECT1) Field Description

Bit	Name	Value	Description
31–5	Reserved		Reads return 0 and writes have no effect.
5–0	INTVECT1[4:0]	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 18-4 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).**

18.14.11SCI Format Control Register (SCIFORMAT)

Figure 18-38 and Table 18-21 illustrate this register.

Figure 18-38. SCI Format Control Register (SCIFORMAT) [offset = 0x28

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													LENGTH[2:0]		
R-0													R/W-0		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													CHAR[2:0]		
R-0													R/WC-0		

R = Read in all modes; W = Write in all modes; WC = Write in SCI-compatible mode only; *-n* = Value after reset

Table 18-21. SCI Format Control Register (SCIFORMAT) Field Description

Bit	Name	Value	Description
31–19	Reserved		Reads return 0 and writes have no effect.
18–16	LENGTH[2:0]	0–3h	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.
		000	The response field has 1 byte/character.
		001	The response field has 2 bytes/characters.
		010	The response field has 3 bytes/characters.
		011	The response field has 4 bytes/characters.
		100	The response field has 5 bytes/characters.
		101	The response field has 6 bytes/characters.
		110	The response field has 7 bytes/characters.
		111	The response field has 8 bytes/characters.
15–3	Reserved		Reads return 0 and writes have no effect.

Table 18-21. SCI Format Control Register (SCIFORMAT) Field Description (Continued)

Bit	Name	Value	Description
2–0	CHAR[2:0]	0–7h	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.**
		000	The character is 1 bit long.
		001	The character is 2 bits long.
		010	The character is 3 bits long.
		011	The character is 4 bits long.
		100	The character is 5 bits long.
		101	The character is 6 bits long.
		110	The character is 7 bits long.
		111	The character is 8 bits long.

18.14.12Baud Rate Selection Register (BRS)

This section describes the baud rate selection register. Figure 18-39 and Table 18-22 illustrate this register.

Figure 18-39. Baud Rate Selection Register (BRS) [offset = 0x2C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	U[2:0]			M[3:0]				PRESCALER P [23:16]							
R-0	R/W-0			R/W-0				R/W-0							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PRESCALER P [15:0]															
R/W-0															

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 18-22. Baud Rate Selection Register (BRS) Field Description

Bit	Name	Value	Description
31	Reserved		This bit is always read as 0. Writes have no effect.
30–28	U[2:0]	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 superfine tuning of the fractional baud rate with seven more intermediate values for each of the M fractional divider values. See Section 18.8.4.1 for more details.
27-24	M[3:0]	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 18.8.4.1 for more details.

Table 18-22. Baud Rate Selection Register (BRS) Field Description (Continued)

Bit	Name	Value	Description
23–0	PRESCALER P [23:0]	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 baudrate selection PRESCALER[27:24]. **NOTE: In LIN mode, ONLY the asynchronous mode and baurdrate values are used.** The baud rate can be calculated using the following formulas: *Asynchronous baud value* = $\left\lfloor \frac{\text{VCLK Frequency}}{16\left(P + 1 + \frac{M}{16}\right)} \right\rfloor$ *Isosynchronous baud value* = $\left(\frac{\text{VCLK Frequency}}{P + 1}\right)$ For P = 0, *Asynchronous baud value* = $\left(\frac{\text{VCLK Frequency}}{32}\right)$ *Isosynchronous baud value* = $\left(\frac{\text{VCLK Frequency}}{2}\right)$ Table 18-23 contains comparative baud values for different P values, with VCLK = 50 MHz, for asynchronous mode..

Table 18-23. 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.

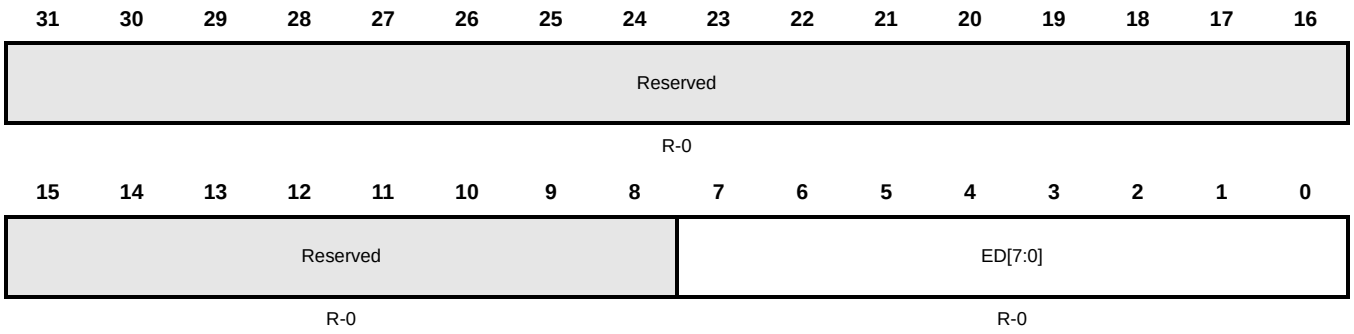
18.14.13SCI Data Buffers (SCIED, SCIRD, SCITD)

The SCI has three addressable registers in which transmit and receive data is stored. These three registers are available in SCI mode only.

18.14.13.1Receiver Emulation Data Buffer (SCIED)

The SCIED register is addressed at a location different from SCIRD, but is physically the same register. Figure 18-40 and Table 18-24 illustrate this register.

Figure 18-40. Receiver Emulation Data Buffer (SCIED) [offset = 0x30



R = Read in all modes; W = Write in SCI-compatible mode only; *-n* = Value after reset

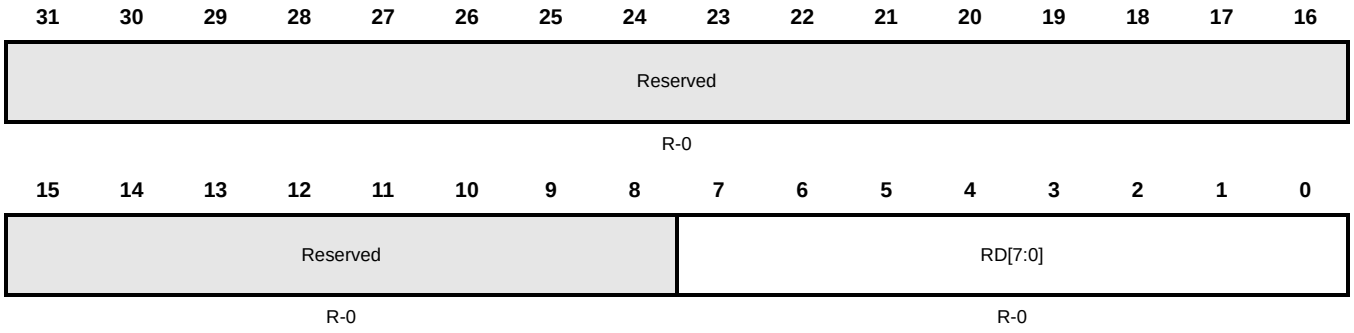
Table 18-24. Receiver Emulation Data Buffer (SCIED) Field Description

Bit	Name	Value	Description
31–8	Reserved		This bit is always read as 0. Writes have no effect.
7–0	ED[7:0]	0–FFh	Emulator data. This bit is effective in compatibility SCI 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.

18.14.13.2Receiver Data Buffer (SCIRD)

This register provides a location for the receiver data. Figure 18-41 and Table 18-25 illustrate this register.

Figure 18-41. Receiver Data Buffer (SCIRD) [offset = 0x34



R = Read in all modes; W = Write in SCI-compatible mode only; *-n* = Value after reset

Table 18-25. Receiver Data Buffer (SCIRD) Field Description

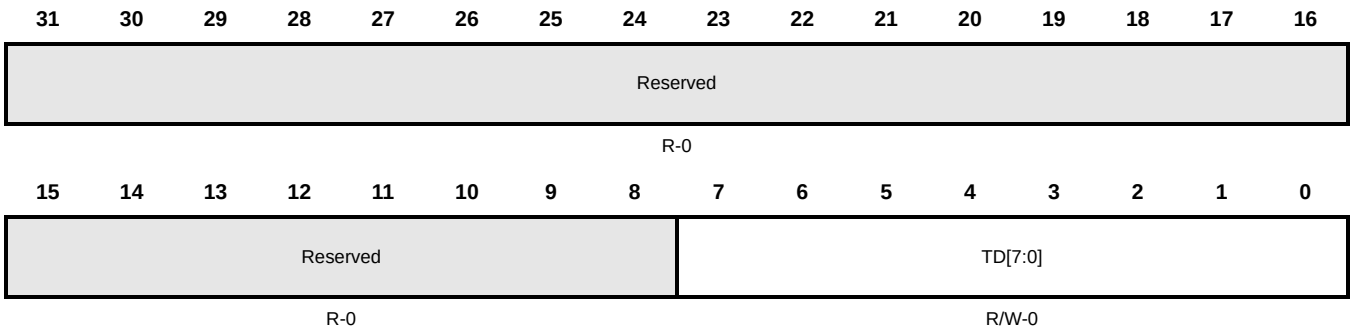
Bit	Name	Value	Description
31–8	Reserved		This bit is always read as 0. Writes have no effect.
7–0	RD[7:0]		Receiver data. This bit is effective in compatibility SCI mode only. When a frame has been completely received, the data in the frame is transferred from the receiver shift register SCIRXSHF to 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, the user software should perform a logical shift on the data by the correct number of positions to make it right justified.

18.14.13.3Transmit Data Buffer Register (SCITD)

Data to be transmitted is written to the SCITD register. Figure 18-42 and Table 18-26 illustrate this register.

Figure 18-42. Transmit Data Buffer Register (SCITD) [offset = 0x38



R = Read in all modes; W = Write in SCI-compatible mode only; *-n* = Value after reset

Table 18-26. Transmit Data Buffer Register (SCITD) Field Description

Bit	Name	Value	Description
31–8	Reserved		This bit is always read as 0. Writes have no effect.
7–0	TD[7:0]	0–FFh	Transmit data. This bit is effective in compatibility SCI 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 TX INT ENA 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.

18.14.14SCI Pin I/O Control Register 0 (SCIPIO0)

Figure 18-43 and Table 18-27 illustrate this register.

Figure 18-43. SCI Pin I/O Control Register 0 (SCIPIO0) [offset = 0x3C

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													TX FUNC	RX FUNC	CLK FUNC
R-0													R/W-0	R/W-0	R/W-0

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 18-27. SCI Pin I/O Control Register 0 (SCIPIO0) Field Description

Bit	Name	Value	Description
31–3	Reserved		This bit is always read as 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	CLK FUNC		Clock function. This bit is effective in LIN or SCI mode. This bit defines the function of pin SCICLK. In LIN mode, the SCICLK pin can only be a digital I/O pin.
		0	SCICLK is a general-purpose digital I/O pin.
		1	SCICLK is the SCI serial clock pin.

18.14.15SCI Pin I/O Control Register 1 (SCIPIO1)

Figure 18-44 and Table 18-28 illustrate this register.

Figure 18-44. SCI Pin I/O Control Register 1 (SCIPIO1) [offset = 0x40

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													TX DIR	RX DIR	CLK DIR
R-0													R/W-0	R/W-0	R/W-0

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 18-28. SCI Pin I/O Control Register 1 (SCIPIO1)

Bit	Name	Value	Description
31–3	Reserved		Reads return 0 and 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 18-29 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 18-30 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.

Table 18-28. SCI Pin I/O Control Register 1 (SCIPIO1)

Bit	Name	Value	Description
0	CLK DIR		Clock pin direction. This bit is effective in LIN or SCI mode. This bit determines the data direction on the SCICLK pin. The direction is defined differently depending upon the value of the CLK FUNC bit. If CLK FUNC = 0 (SCICLK = general-purpose I/O), then the CLK DIR bit has the following function:
		0	SCICLK pin is a general-purpose input pin.
		1	SCICLK pin is a general-purpose output pin.
			If CLK FUNC = 1 (SCICLK pin has SCI clock functionality) and an internal clock has been selected then the CLK DIR bit has the following functionality, in SCI compatibility mode only:
		0	The internal SCI clock is not output on SCICLK pin.
		1	The internal SCI clock is output on SCICLK pin.
			If CLK FUNC = 1 (SCICLK pin has SCI clock functionality) and an external clock has been selected (SCIGCR1[5] = 0), then the SCI expects an external clock signal not to exceed VCLK/16 on the SCICLK pin (see Table 18-31 for a description of the behavior of the pin with different bits set).

Table 18-29. 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 LINTX pin.

Table 18-30. 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 LINRX pin.

Table 18-31. SCICLK Pin Control

Function	CLK IN[1]	CLK OUT	CLK FUNC	CLK DIR
SCICLK (output)	X	X	1	1
SCICLK (input)	X	X	1	0
General purpose input	X	X	0	0
General purpose output, high	X	1	0	1
General purpose output, low	X	0	0	1

1 CLK IN is a read-only bit. Its value always reflects the level of the SCICLK pin.

18.14.16SCI Pin I/O Control Register 2 (SCIPIO2)

Figure 18-45 and Table 18-32 illustrate this register.

Figure 18-45. SCI Pin I/O Control Register 2 (SCIPIO2) [offset = 0x44

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													TX IN	RX IN	CLK IN
R-0													R-X	R-X	R-X

R = Read in all modes; -*n* = Value after reset; -*x* = Indeterminate

Table 18-32. SCI Pin I/O Control Register 2 (SCIPIO2) Field Description

Bit	Name	Value	Description
31–3	Reserved		Reads return 0 and 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	CLK IN		Clock pin in. This bit is effective in LIN or SCI mode. This bit contains the current value on the SCICLK pin.
		0	The SCICLK pin is at logic low (0).
		1	The SCICLK pin is at logic high (1).

18.14.17SCI Pin I/O Control Register 3 (SCIPIO3)

Figure 18-46 and Table 18-33 illustrate this register.

Figure 18-46. SCI Pin I/O Control Register 3 (SCIPIO3) [offset =0x48

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													TX OUT	RX OUT	CLK OUT
R-0													R/W-0	R/W-0	R/W-0

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 18-33. SCI Pin I/O Control Register 3 (SCIPIO3) Field Description

Bit	Name	Value	Description
31–3	Reserved		Reads return 0 and writes have no effect.
2	TX OUT		Transmit pin out. This bit is effective in LIN or SCI mode. This pin 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 18-29 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 SC 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 18-30 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)

Table 18-33. SCI Pin I/O Control Register 3 (SCIPIO3) Field Description (Continued)

Bit	Name	Value	Description
0	CLK OUT		Clock data out. This bit is effective in LIN or SCI mode. This bit contains the data to be output on pin SCICLK if the following conditions are met: • CLK FUNC = 0 (SCICLK pin is a general-purpose I/O.) • CLK DIR = 1 (SCICLK pin is a general-purpose output.) See Table 18-31 for an explanation of this bit's effect in combination with other bits.
		0	Output value on SCICLK is a 0 (logic low).
		1	Output value on SCICLK is a 1 (logic high). (Output voltage is V_{OH} or higher if CLKPDR = 0 and output is in high impedance state if CLKPDR = 1)

18.14.18SCI Pin I/O Control Register 4 (SCIPIO4)

Figure 18-47 and Table 18-34 illustrate this register.

Figure 18-47. SCI Pin I/O Control Register 4 (SCIPIO4) [offset =0x4C .

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													TX SET	RX SET	CLK SET
R-0													R/W-0	R/W-0	R/W-0

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 18-34. SCI Pin I/O Control Register 4 (SCIPIO4) Field Description

Bit	Name	Value	Description
31–3	Reserved		Reads return 0 and 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 18-29 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 data 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 18-30 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).

Table 18-34. SCI Pin I/O Control Register 4 (SCIPIO4) Field Description (Continued)

Bit	Name	Value	Description
0	CLK SET		Clock pin set. This bit is effective in LIN or SCI mode. This bit sets the data to be output on pin SCICLK if the following conditions are met: • CLK FUNC = 0 (SCICLK pin is a general-purpose I/O.) • CLK DIR = 1 (SCICLK pin is a general-purpose output.) See Table 18-31 for an explanation of this bit's effect in combination with the other bits.
		0	*Read:* The output on SCICLK is at logic low (0). *Write:* Writing a 0 to this bit has no effect.
		1	*Read and write:* The output on SCICLK is at logic high (1).

18.14.19SCI Pin I/O Control Register 5 (SCIPIO5)

Figure 18-48 and Table 18-35 illustrate this register.

Figure 18-48. SCI Pin I/O Control Register 5 (SCIPIO5) [offset = 0x50

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													TX CLR	RX CLR	CLK CLR
R-0													R/W-0	R/W-0	R/W-0

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 18-35. SCI Pin I/O Control Register 5 (SCIPIO5) Field Description

Bit	Name	Value	Description
31–3	Reserved		Reads return 0 and 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).

Table 18-35. SCI Pin I/O Control Register 5 (SCIPIO5) Field Description (Continued)

Bit	Name	Value	Description
0	CLK CLR		Clock pin clear. This bit is effective in LIN or SCI mode. This bit clears the logic to be output on pin SCICLK if the following conditions are met: • CLK FUNC = 0 (SCICLK pin is a general-purpose I/O.) • CLK DIR = 1 (SCICLK pin is a general-purpose output.)
		0	*Read:* The output on SCICLK is at logic low (0). *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The output on SCICLK is at logic high (1). *Write:* The output on SCICLK is at logic low (0).

18.14.20SCI Pin I/O Control Register 6 (SCIPIO6)

Figure 18-49 and Table 18-36 illustrate this register.

Figure 18-49. SCI Pin I/O Control Register 6 (SCIPIO6) [offset = 0x54

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													TX PDR	RX PDR	CLK PDR
R-0													R/W-0	R/W-0	R/W-0

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 18-36. SCI Pin I/O Control Register 6 (SCIPIO6) Field Description

Bit	Name	Value	Description
31–3	Reserved		Reads return 0 and writes have no effect.
2	TX PDR		Transmit pin open drain enable. This bit is effective in LIN or SC 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 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 train enable. This bit is effective in LIN or SC 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 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.

Table 18-36. SCI Pin I/O Control Register 6 (SCIPIO6) Field Description (Continued)

Bit	Name	Value	Description
0	CLK PDR		Clock pin open drain enable. This bit is effective in LIN or SCI mode. This bit enables open-drain capability in the output pin SCICLK if the following conditions are met: • CLK FUNC = 0 (SCICLK pin is a general-purpose I/O.) • CLK DIR = 1 (SCICLK pin is a general-purpose output.)
		0	Open drain functionality is disabled; the output voltage is V_{OL} or lower if CLKOUT =0 and V_{OH} or higher if CLKOUT =1.
		1	Open drain functionality is enabled; the output voltage is V_{OL} or lower if CLKOUT =0 and high impedance if CLKOUT =1.

18.14.21SCI Pin I/O Control Register 7 (SCIPIO7)

Figure 18-50 and Table 18-37 illustrate this register.

Figure 18-50. SCI Pin I/O Control Register 7 (SCIPIO7) [offset = 0x58

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													TX PD	RX PD	CLK PD
R-0													R/W-n	R/W-n	R/W-n

R = Read in all modes; W = Write in all modes; -*n* = Value after reset, Refer to the Terminal Functions in the device datasheet for default pin settings.

Table 18-37. SCI Pin I/O Control Register 7 (SCIPIO7) Field Description

Bit	Name	Value	Description
31–3	Reserved		Reads return 0 and 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	CLK PD		Clock pin pull control disable. This bit is effective in LIN or SCI mode. This bit disables the pull control capability on the input pin SCICLK.
		0	Pull control on the SCICLK pin is enabled.
		1	Pull control on the SCICLK pin is disabled.

18.14.22SCI Pin I/O Control Register 8 (SCIPIO8)

Figure 18-51 and Table 18-38 illustrate this register.

Figure 18-51. SCI Pin I/O Control Register 8 (SCIPIO8) [offset = 0x5C

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													TX PSL	RX PSL	CLK PSL
R-0													R/W-*n*	R/W-*n*	R/W-*n*

R = Read in all modes; W = Write in all modes; -*n* = Value after reset, Refer to the Terminal Functions in the device datasheet for default pin settings.

Table 18-38. SCI Pin I/O Control Register 8 (SCIPIO8) Field Description

Bit	Name	Value	Description
31–3	Reserved		Reads return 0 and writes have no effect.
2	TX PSL		TX 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		RX 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	CLK PSL		CLK pin pull select. This bit is effective in LIN or SCI mode. It selects pull type in the input pin SCICLK.
		0	Pull control on the SCICLK pin is enabled.
		1	Pull control on the SCICLK pin is disabled.

18.14.23LIN Compare Register (LINCOMPARE)

Figure 18-52 and Table 18-39 illustrate this register.

Figure 18-52. LIN Compare Register (LINCOMPARE) [offset = 0x60

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						SDEL[1:0]		Reserved					SBREAK[2:0]		
R-0						RWP-0		R-0					RWP-0		

R = Read in all modes; WL = Write in LIN mode only; RWP = Read/Write in privileged mode only; *-n* = Value after reset

Table 18-39. LIN Compare Register (LINCOMPARE) Field Description

Bit	Name	Value	Description
31–10	Reserved		Reads return 0 and writes have no effect.
9–8	SDEL[1:0]		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 0x00. The formula to program the value (in T_{bits}) for the synchronization delimiter is $T_{SDEL} = (SDEL + 1)T_{bit}$
		00	The synch delimiter has 1 T_{bit}.
		01	The synch delimiter has 2 T_{bit}.
		10	The synch delimiter has 3 T_{bit}.
		11	The synch delimiter has 4 T_{bit}.
7-3	Reserved		Reads return 0 and writes have no effect.

Table 18-39. LIN Compare Register (LINCOMPARE) Field Description (Continued)

Bit	Name	Value	Description
2–0	SBREAK[2:0]		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 0x0, which adds nothing to the automatically generated SYNCH BREAK.** The formula to program the value (in T_{bits}) for the SYNCH BREAK is $T_{SYNBRK} = 13T_{bit} + SBREAK \times T_{bit}$
		000	The synch break has no additional T_{bit}.
		001	The synch break has 1 additional T_{bit}.
		010	The synch break has 2 additional T_{bit}.
		011	The synch break has 3 additional T_{bit}.
		100	The synch break has 4 additional T_{bit}.
		101	The synch break has 5 additional T_{bit}.
		110	The synch break has 6 additional T_{bit}.
		111	The synch break has 7 additional T_{bit}.

18.14.24LIN Receive Buffer 0 Register (LINRD0)

Figure 18-53 and Table 18-40 illustrate this register.

Figure 18-53. LIN Receive Buffer 0 Register (LINRD0) [offset = 0x64

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RD0[7:0]								RD1[7:0]							
R-0								R-0							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RD2[7:0]								RD3[7:0]							
R-0								R-0							

R = Read in all modes; -*n* = Value after reset

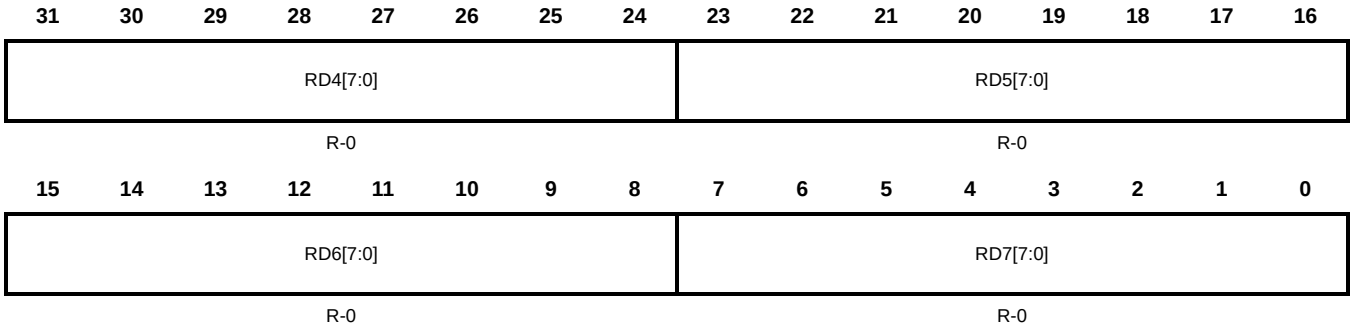
Table 18-40. LIN Receive Buffer 0 Register (LINRD0) Field Description

Bit	Name	Value	Description
31–24	RD0[7:0]	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[7:0]	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[7:0]	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[7:0]	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.

18.14.25LIN Receive Buffer 1 Register (LINRD1)

Figure 18-54 and Table 18-41 illustrate this register.

Figure 18-54. LIN Receive Buffer 1 Register (RD1) [offset = 0x68]



R = Read in all modes; -*n* = Value after reset

Table 18-41. LIN Receive Buffer 1 Register (RD1) Field Description

Bit	Name	Value	Description
31–24	RD4[7:0]	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[7:0]	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[7:0]	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[7:0]	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.

18.14.26LIN Mask Register (LINMASK)

Figure 18-55 and Table 18-42 illustrate this register.

Figure 18-55. LIN Mask Register (LINMASK) [offset = 0x6C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								RX ID MASK[7:0]							
R-0								R/WL-0							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TX ID MASK[7:0]							
R-0								R/WL-0							

R = Read in all modes; WL = Write in LIN mode only; *-n* = Value after reset

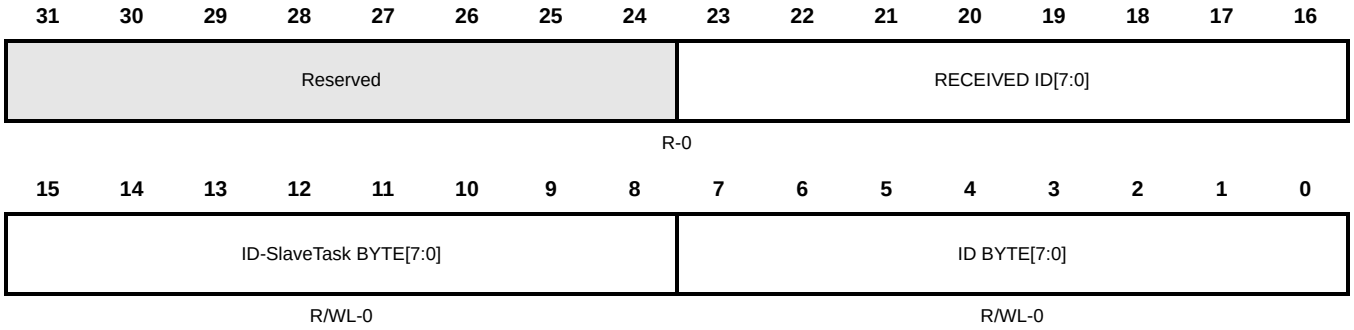
Table 18-42. LIN Mask Register (LINMASK) Field Description

Bit	Name	Value	Description
31–24	Reserved		Reads of these bits return 0 and writes have no effect.
23–16	RX ID MASK[7:0]	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		Reads of these bits return 0 and writes have no effect.
7–0	TX ID MASK[7:0]	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.

18.14.27LIN Identification Register (LINID)

Figure 18-56 and Table 18-43 illustrate this register.

Figure 18-56. LIN Identification Register (LINID) [offset = 0x70



R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 18-43. LIN Identification Register (LINID) Field Description

Bit	Name	Value	Description
31–24	Reserved		Reads return 0 and writes have no effect.
23–16	Received ID[7:0]	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[7:0]	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[7:0]	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 HGENCTRL = 0.

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 0xFF and the TX ID MASK must be set to 0xFF.

18.14.28LIN Transmit Buffer 0 Register(LINTD0)

Figure 18-57 and Table 18-44 illustrate this register.

Figure 18-57. LIN Transmit Buffer 0 Register (LINTD0) [offset = 0x74

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TD0[7:0]								TD1[7:0]							
R/W-0								R/W-0							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TD2[7:0]								TD3[7:0]							
R/W-0								R/W-0							

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 18-44. LIN Transmit Buffer 0 Register (LINTD0) Field Description

Bit	Name	Value	Description
31-24	TD0[7:0]	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[7:0]	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[7:0]	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[7:0]	0–FFh	8-Bit transmit buffer 3. Byte 3 to be transmitted is written into this register and then copied to SCITXSHF for transmission.

18.14.29LIN Transmit Buffer 1 Register (LINTD1)

Figure 18-58 and Table 18-45 illustrate this register.

Figure 18-58. LIN Transmit Buffer 1 Register (LINTD1) [offset = 0x78

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TD4[7:0]								TD5[7:0]							
R/W-0								R/W-0							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TD6[7:0]								TD7[7:0]							
R/W-0								R/W-0							

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 18-45. LIN Transmit Buffer 1 Register (LINTD1) Field Description

Bit	Name	Value	Description
31-24	TD4[7:0]	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[7:0]	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[7:0]	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[7:0]	0–FFh	8-Bit transmit buffer 7. Byte 7 to be transmitted is written into this register and then copied to SCITXSHF for transmission.

18.14.30Maximum Baud Rate Selection Register (MBRS)

Figure 18-59 and Table 18-46 illustrate this register.

Figure 18-59. Maximum Baud Rate Selection Register (MBRS) [offset = 0x7C

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			MBR[12:0]												
R-0			R/WL-0	R/WL-1	R/WL-1	R/WL-0	R/WL-1	R/WL-1	R/WL-0	R/WL-1	R/WL-0	R/WL-1	R/WL-1	R/WL-0	R/WL-0

R = Read in all modes; WL = Write in LIN mode only; *-n* = Value after reset

Table 18-46. Maximum Baud Rate Selection Register (MBRS) Field Description

Bit	Name	Value	Description
31–13	Reserved		Reads return 0 and writes have no effect.
12–0	MBR[12:0]	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 18.8.5.2) 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 mistakingly 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. $MBR = \frac{0.9 \times VCLK}{maxbaudrate}$

18.14.31Input/Output Error Enable (IODFTCTRL) Register

All the bits in the IODFTCTRL register are used in IODFT (I/O design for test) mode only. Figure 18-60 and Table 18-47 illustrate this register. After the basic SCI/LIN module configuration, enable the required Error mode to be creeated followed by IODFT Key enable.

Figure 18-60. Input/Output Error Enable Register (IODFTCTRL) [offset = 0x90]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BEN	PBEN	CEN	ISFE	Reserved	FEN	PEN	BRKDT ENA	Reserved			PIN SAMPLE MASK[1:0]		TX SHIFT [2:0]		
R/WL-0	R/WL-0	R/WL-0	R/WL-0	R-0	R/W-0	R/WC-0	R/WC-0	R-0			R/W-0	R/W-0	R/W-0		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				IODFTENA[3:0]				Reserved						LPB ENA	RXP ENA
R-0				R/WP-0	R/WP-1	R/WP-0	R/WP-1	R-0						R/WP-0	R/WP-0

R = Read in all modes; W = Write in all modes; WL = Write in LIN mode only; WC = Write in sci-compatible mode only; WP = Write in privilege mode only.
-*n* = Value after reset

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

Table 18-47. Input/Output Error Enable Register (IODFTCTRL) Field Description

Bit	Name	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.

Table 18-47. Input/Output Error Enable Register (IODFTCTRL) Field Description (Continued)

Bit	Name	Value	Description
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		Reads return 0 and 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 effective in compatibility 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	BRKD TENA		Break detect error enable. This bit is effective in SCI-compatibility 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_{BITS} so that a BRKDT error occurs.
23–21	Reserved		Reads return 0 and writes have no effect.

Table 18-47. Input/Output Error Enable Register (IODFTCTRL) Field Description (Continued)

Bit	Name	Value	Description
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).**
		00	No mask is used.
		01	Invert the TX Pin value at TBIT_CENTER.
		10	Invert the TX Pin value at TBIT_CENTER + SCLK.
		11	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.
		000	No delay occurs.
		001	The value is delayed by 1 SCLK.
		010	The value is delayed by 2 SCLK.
		011	The value is delayed by 3 SCLK.
		100	The value is delayed by 4 SCLK.
		101	The value is delayed by 5 SCLK.
		110	The value is delayed by 6 SCLK.
		111	The value is delayed by 7 SCLK.
15–12	Reserved		Reads return 0 and writes have no effect.
11–8	IODFTENA		IO DFT enable key. Write access permitted in Priviledge mode only/
		1010	IODFT is enabled.
		All other values	IODFT is disabled.
7–2	Reserved		Reads return 0 and writes have no effect.
1	LPBENA		Module loopback enable. Write access permitted in Priviledge mode only. **Note: In analog loopback mode the complete communication path through the I/Os can be tested, whereas in digital loop-back 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.

Table 18-47. Input/Output Error Enable Register (IODFTCTRL) Field Description (Continued)

Bit	Name	Value	Description
0	RXPENA		Module analog loopback through receive pin enable.Write access permitted in Priviledge 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 loop-back mode)
		0	Analog loopback through the transmit pin is enabled.
		1	Analog loopback through the receive pin is enabled.

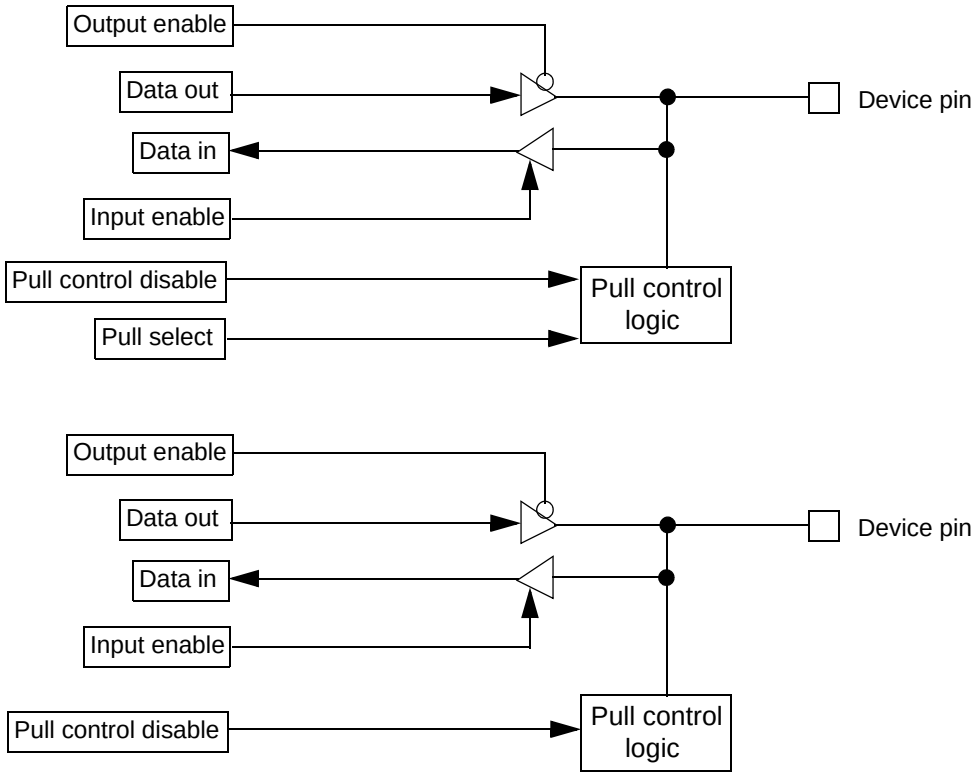
18.15 GPIO Functionality

The following sections apply to all device pins that can be configured as functional or general-purpose I/O pins.

18.15.1 GPIO Functionality

Figure 18-61 illustrates the GPIO functionality.

Figure 18-61. GPIO Functionality



18.15.2 Under Reset

The following apply if a device is under reset:

- Pull control. The reset pull control on the pins is enabled or disabled depending on a device-specific option. This feature is configurable for each module separately.
- Input buffer. If the reset pull control is enabled, then the input buffer is also enabled. If the reset pull control is disabled then the input buffer is also disabled.
- Output buffer. The output buffer is disabled.

18.15.3 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 18.14.21). In this case, if the PSL (pull select) bit in the SCIPIO8 register (Section 18.14.22) 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 disabled only if the pin direction is set as input in the SCIPIO1 register (Section 18.14.15) AND the pull control is disabledAND pull down is selected as the pull bias (the PSL bit in the SCIPIO8 register; Section 18.14.22). In all other cases, the input buffer is enabled.

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 18.14.21).

- 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 18.14.15) AND the open-drain feature is not enabled in the SCIPIO6 register (Section 18.14.20)

18.15.4 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 18.14.14).

18.15.5 Summary

The behavior of the input buffer, output buffer, and the pull control is summarized in Table 18-48.

Table 18-48. 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	Device- and module-specific	Disabled	Depends on pull control
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
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

General-Purpose Input/Output (GIO) Module

The general-purpose input/output (GIO) module provides the TMS470M Series of microcontrollers with input/output (I/O) capability. The I/O pins are bidirectional and bit-programmable. The GIO module also supports external interrupt capability on up to 32 pins. Please check the device datasheet for the number of GIO pins in the device and the pins with interrupt capability.

Topic		Page
19.1	Overview	1052
19.2	Functional Description of GIO Module	1053
19.3	Device Modes of Operation	1060
19.4	Pullup/Pulldown Function	1061
19.5	Open Drain Function	1062
19.6	GIO Control Registers	1063
19.7	Applications	1092

19.1 Overview

The GIO module has the following features:

- Each I/O pin is controlled by bits in these registers:
  - Data direction (GIODIR; Section 19.6.11)
  - Data input (GIODIN; Section 19.6.12)
  - Data output (GIODOUT; Section 19.6.13)
  - Data set (GIODSET; Section 19.6.14)
  - Data clear (GIODCLR; Section 19.6.15)
  - Open drain (GIOPDR; Section 19.6.16)
  - Pull disable (GIOPULDIS; Section 19.6.17)
  - Pull select (GIOPSL; Section 19.6.18)
- The interrupts have the following characteristics:
  - Programmable interrupt detection either on both edges or on a single edge (set in GIOINTDET; Section 19.6.2)
  - Programmable edge-detection polarity, either rising or falling edge (set in GIOPOL register; Section 19.6.11)
  - Individual interrupt flags (set in GIOFLG register; Section 19.6.6)
  - Individual interrupt enables, set and cleared through GIOENASET and GIOENACLR registers respectively (Section 19.6.4)
  - Programmable interrupt priority, set through GIOLVLSET and GIOLVLCLR registers (Section 19.6.5)
- Internal pullup/pulldown allows unused I/O pins to be left unconnected.

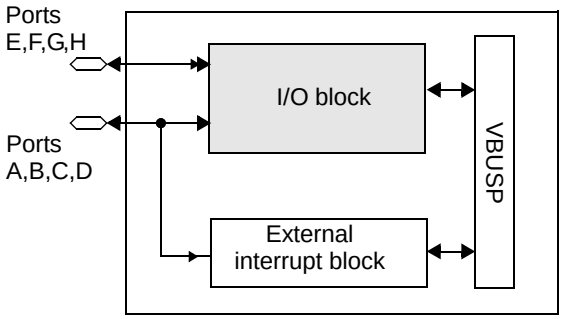
19.2 Functional Description of GIO Module

The GIO module (see Figure 19-1) comprises two separate components: an input/output (I/O) block and an external interrupt block. The GIO module can have up to 8 ports (A through H) with up to 8 pins (0 through 7) per port. The pins for ports E, F, G, and H handle the standard I/O functions and are connected directly to the VBUSP; see Figure 19-1. Ports A, B, C, and D can handle interrupts in addition to the standard I/O functions.

Note:
The number of GIO ports, the number of pins per port, and the pins with interrupt capability are device specific. Please check the device datasheet.

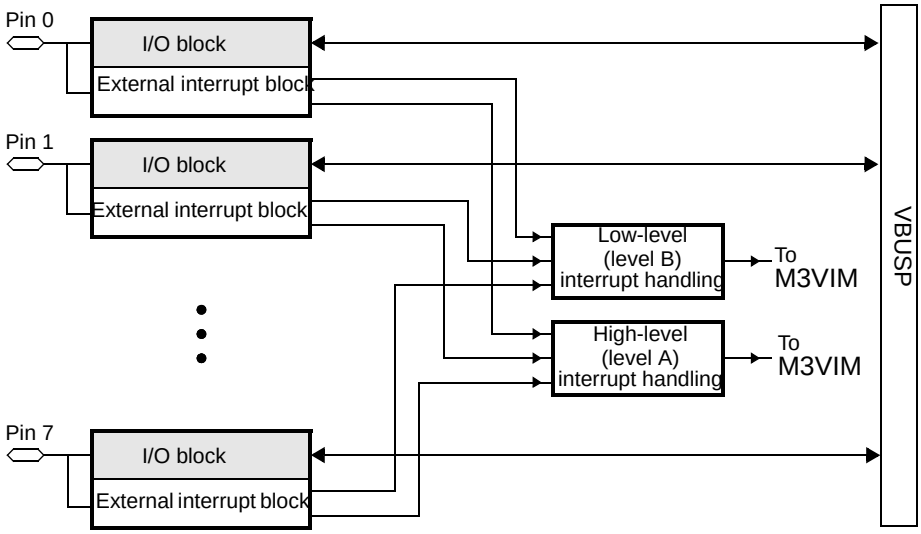
Note:
The device datasheet may specify the GIO ports as A, B, C, D etc., or as P0, P1, P2, P3 etc. Port A is equivalent to Port P0, Port B to Port P1 and so on.

Figure 19-1. GIO Module Diagram



The pins on port A (and optionally port B, C and D), shown in Figure 19-2, are all interrupt-capable pins and can be used to handle either general I/O functions or external interrupt signals. The pins are connected to both an I/O block and an external interrupt block. Each of the eight I/O blocks is connected to the VBUSP bus, whereas the interrupt blocks are physically attached to a single high-level-interrupt-handling block and to a single low-level-interrupt-handling block. The high-level-interrupt-handling block, which is also denoted as level A, and the low-level-interrupt-handling block, which is also denoted as level B, each send one signal to the vectored interrupt manager (M3VIM) in the system module for processing. The interrupt priority of level A and level B interrupt handling blocks can be re-programmed in the M3VIM.

Figure 19-2. GIO Port A / B / C / D Module Diagram



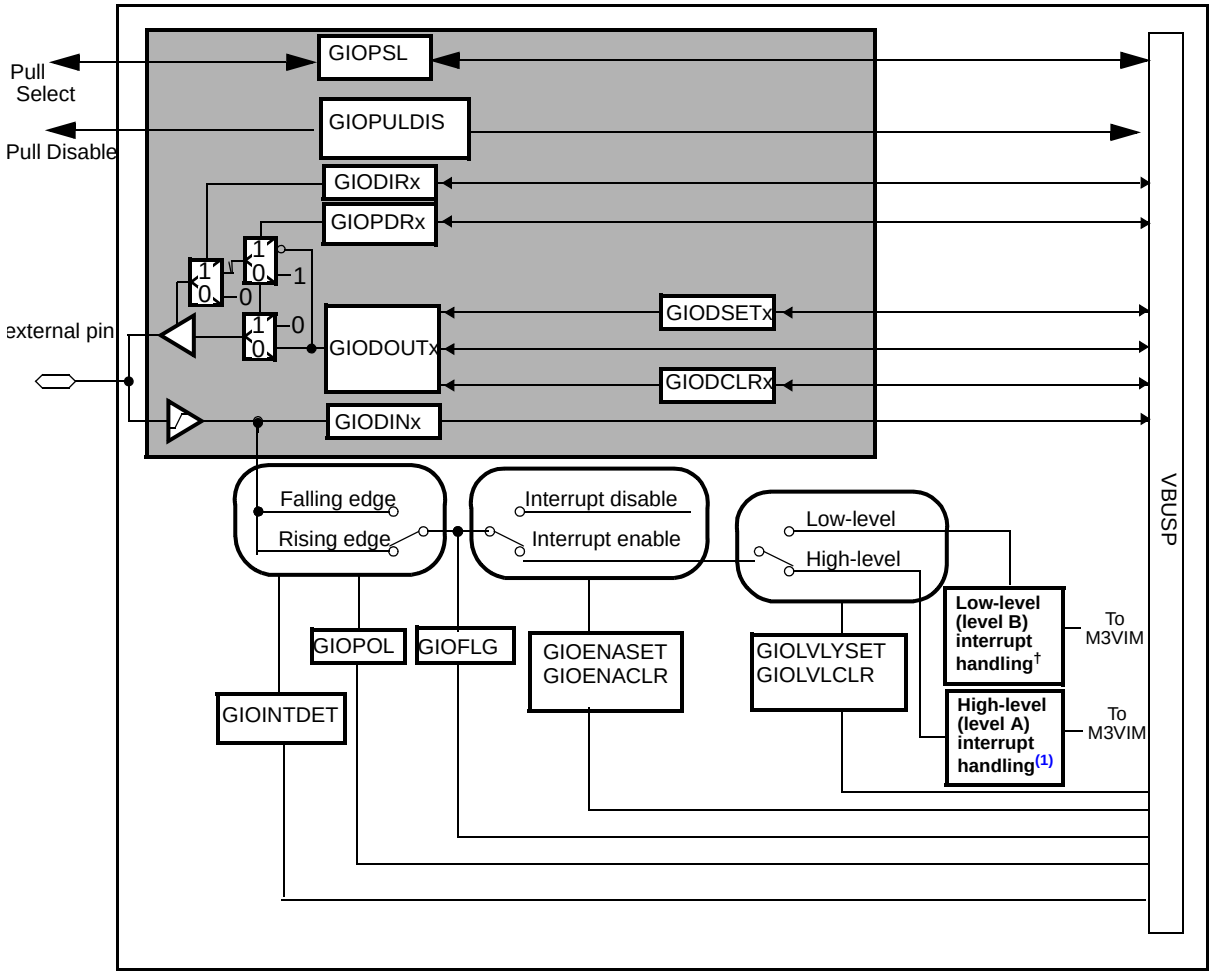
1 Not all devices have 32 external interrupts. The exact number of external interrupts is device specific; see the data sheet in for details.

19.2.1 GIO Block Diagram

The GIO block diagram (see Figure 19-3) represents the flow of information through a pin. The shaded area corresponds to the I/O block; the unshaded area corresponds to the external interrupt block.

Because ports E, F, G, and H are not interrupt-capable, the block diagram for those ports reduces to the shaded portion of the block diagram in Figure 19-3.

Figure 19-3. 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.

19.2.2 I/O Blocks

Each pin serviced by port A, B, C, D, E, F, G, and H contains its individual I/O block.

19.2.2.1 I/O Function

The GIO module sends data to the external pin through the output buffer and receives data from the external pin through the input buffer (see Figure 19-3). The associated registers are:

- GIODIR[7:0] (Section 19.6.11)—Controls the direction that information is sent. The GIODIR[7:0] register determines whether or not values in the data output register are sent to the external pin. The input buffer is enabled except in the case when the pin direction is set as an input (GIODIR[7:0]) AND the pull control is disabled (GIOPULDIS[7:0]) AND pull down is selected as the pull bias (GIOPSL[7:0]). Refer to Table 19-4. When the input buffer is enabled, information that is sent to the external pin is also received in the input buffer. GIODOUT[7:0] (Section 19.6.13)—Controls what information is sent to the external pin when it is configured as an output. When the output buffer is enabled, writing values to the data output register

(GIODOUT[7:0]) applies a voltage to the output pin. 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 forces the pin to a high output voltage (V_{OH} or higher) if the open drain functionality is disabled (GIOPDR[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).

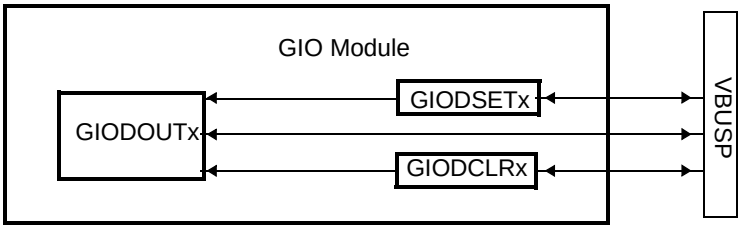
- GIODIN[7:0] (Section 19.6.12)—Receives the information from the external pin. A high voltage (V_{IH} or greater) applied to the pin causes a high value (1) in the data input register (GIODIN[7:0]). When a low voltage (V_{IL} or less) is applied to the pin, the data input register reads a low value (0). The V_{IH} and V_{IL} values are device specific and can be found in the device datasheet.
- GIOPDR[7:0] (Section 19.6.16)—Controls the open drain configuration of the pin. If the pin is set as open drain, a high value (1) written to the data output register (GIODOUT[7:0]) forces the pin to a high impedance state (z). Open drain functionality is enabled or disabled using the open drain register GIOPDR[7:0].
- GIOPULDIS[7:0] (Section 19.6.17)—Disables the pull control capability at pin. The pull functionality for the pin can be enabled or disabled in the GIOPULDIS[7:0] register.
- GIOPSL[7:0] (Section 19.6.18)—Selects the pull type at pin. Pull down or pull up can be selected using the GIOPSL[7:0] register.

19.2.2.2 Output Control Registers

When a GIO pin is configured as an output pin, the value in the data output register (GIODOUT[7:0]; Section 19.6.13) specifies the voltage applied to the external pin. The GIO module provides three ways of communicating with the data output register (see Figure 19-4).

- The control register bit can be written directly by writing to the data output register (GIODOUT[7:0]; Section 19.6.13).
- The data output register bit can be set to 1 using the data set register (GIODSET[7:0]; Section 19.6.14).
- The data output register bit can be cleared to 0 using the data clear register (GIODCLR[7:0]; Section 19.6.15).

Figure 19-4. Communication With the Data Output Register



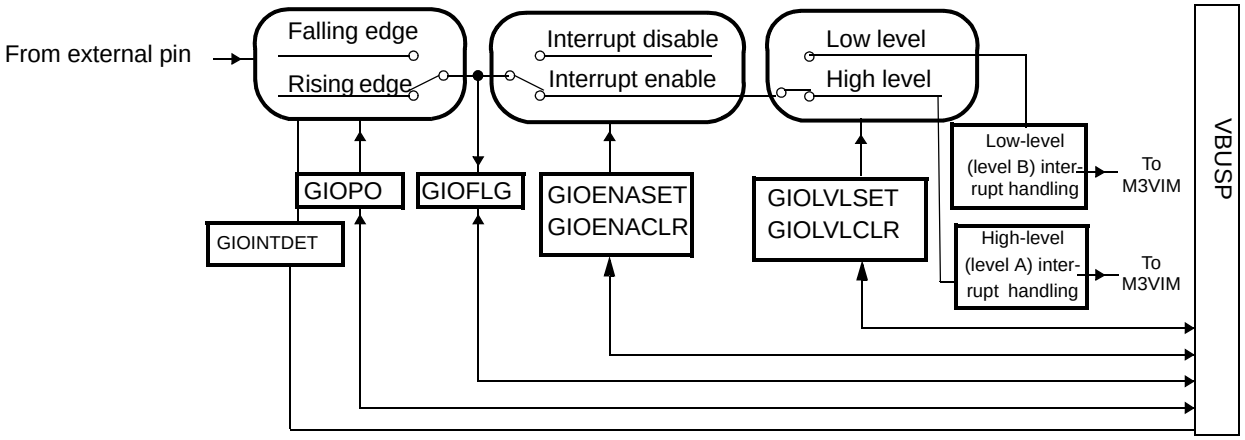
The GIODSET[7:0] and GIODCLR[7:0] registers allow improved handling of data. The data set and data clear registers preclude any possibility of a read-modify-write (RMW) operation. RMW operations are possible when the CPU reads a register, performs some action (for example, an OR operation), and then writes the values back into the register. Under such conditions, it is possible that the contents of the original register (GIODOUT[7:0]) can change (for example, an interrupt procedure) between the time when the CPU originally reads the register and the time when the CPU writes the new value. If the contents of the register changes between the time the CPU reads the register and the time it writes the new value to that register, then the CPU has ended up using an outdated register value as the basis for its operations, and therefore generates an erroneous result from those operations that also ends up being written back into the register. RMW operations can be avoided by using the GIODSET[7:0] and GIODCLR[7:0] registers.

19.2.3 External Interrupt Block

Each interrupt-capable pin connects to the GIO module's single low-level-interrupt-handling block and single high-level-interrupt-handling block. Depending on the priority, the interrupt signal is sent through the

appropriate offset register to the vectored interrupt manager (M3VIM) in the system module (see Section 19.2.3.3). Figure 19-5 shows the external interrupt block.

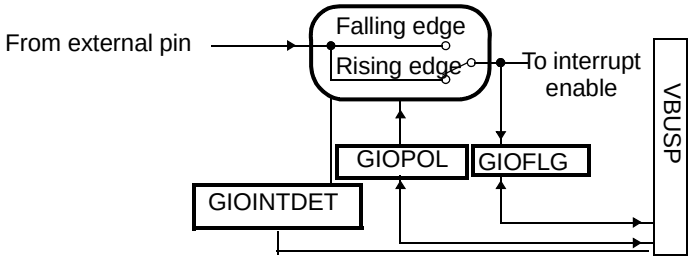
Figure 19-5. External Interrupt Block



19.2.3.1 Edge Detection and the Flag Register

The edge-detection hardware and flag register, like the input buffer, are always enabled. The GIOINTDET register (Section 19.6.2) specifies whether interrupt detection is on both edges or on a single edge only. If interrupt detection is on a single edge only, then the GIOPOL register (see Figure 19-6 and Section 19.6.3) is used to define whether a rising edge or a falling edge is recognized as an interrupt.

Figure 19-6. Edge Detection and Flag Register



A rising edge occurs when the voltage on a given pin transitions from a low value (V_{IL} or lower) to a high value (V_{IH} or higher). The voltage on the external pin must remain at the high level for at least one VCLK cycle to ensure recognition.

A falling edge occurs when the voltage on the external pin transitions from a high value (V_{IH} or higher) to a low value (V_{IL} or lower). The voltage on the external pin must remain at the low level for at least one VCLK cycle to ensure recognition. (GIOPOL behaves differently in a low-power state. See Section 19.3.2 for more information.)

The GIOINTDET register (Section 19.6.2) is used to define whether the interrupt flag is set on both rising and falling edges or on a single edge only. Single edge polarity is defined in the polarity register (GIOPOL; Section 19.6.3).

The corresponding flag in the GIOFLG register (Section 19.6.6) is set when a transition appearing on the external pin matches the combination of edges chosen by the GIOINTDET and GIOPOL registers. For example, to set the flag on only a rising edge on pin 2 of GIO port A, clear the bit in the interrupt detection register (GIOINTDET[2] = 0), and set the bit in the polarity register (GIOPOL[2] = 1). Then, when the signal transition takes place, the GIO module will set the appropriate flag in the flag register (GIOFLG[2] = 1).

Note: Setting Flag With Interrupt Disabled

A flag can be set whether or not the interrupt is enabled. The flag register can then be polled instead of driving an interrupt. Additionally, the flag should not be set before

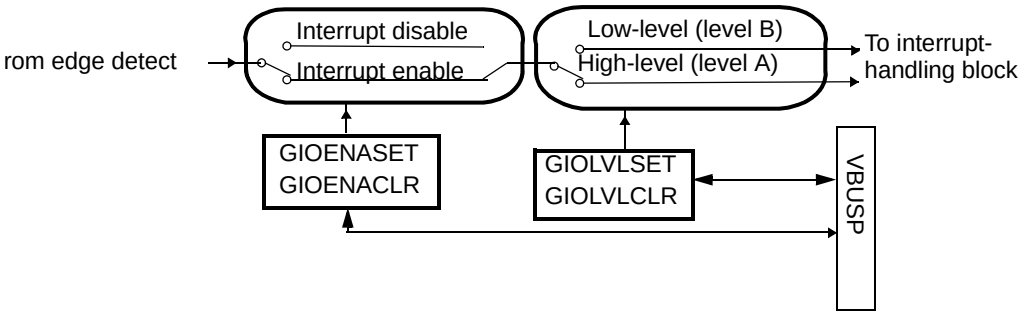
enabling the interrupt; specifically, the flag register should be cleared before enabling the interrupt.

The edge-detection hardware responds to voltages on the external pin and does not discriminate between the source of these voltages. Therefore, the interrupt will respond to the correct edge even when generated from a pin whose output buffer is enabled.

19.2.3.2 Interrupts and Interrupt Levels

The interrupt flag is set when an edge transition on the external pin is detected and matches the edge as chosen by the GIOINTDET and GIOPOL registers. An interrupt can be generated from the set flag if the interrupt is enabled (see Figure 19-7).

Figure 19-7. Interrupt Enable and Priority Level



The external interrupt can be enabled or disabled using the GIOENASET and GIOENACLR register (Section 19.6.4) bits respectively. These two registers are physically implemented as a single register. If the interrupt is enabled, the signal with an appropriate edge leads to an interrupt. If multiple interrupts occur simultaneously, the GIO module must prioritize the interrupts so that they can be handled in the proper order.

The order in which simultaneous GIO interrupts are processed is determined by the following criteria:

1. The GIO priority control registers (GIOLVLSET and GIOLVLCLR; Section 19.6.5) provide a software-implemented prioritization scheme. Each pin can be set as either a high-level (level A) or low-level interrupt (level B). By default in the Vectored Interrupt Manager (M3VIM), interrupts with level A are higher priority than those of level B and therefore are serviced first. However, the priority of level A and level B can be re-programmed in the M3VIM. Refer to the M3VIM module user's guide. The handling of interrupts with the same priority is determined by the interrupt with the lowest bit value having the highest priority. This prioritization is hardwired into the module.

Note: Wakeup Condition
GIOA interrupts are also used to awaken the device from the low power modes. The wakeup interrupt is level-based rather than edge-based.

Table 19-1 shows an example of how interrupt priorities are determined. Four interrupts occur simultaneously on GIO port A pins 3–0, and are handled as following:

1. The interrupts on pin 1 and pin 0 are both sent to the Vectored Interrupt Manager (M3VIM) for servicing because they have the lowest bit values among the high-level and low-level interrupts. Assuming that the M3VIM is set to service level A interrupt priority before level B interrupt priority, pin 1 is serviced first because it is the high-priority interrupt of the two.
2. Next, the interrupts on pin 3 and pin 0 are sent to the M3VIM for servicing. The interrupt on pin 3 is serviced because it is the only remaining high-priority interrupt.
3. The interrupt on pin 0 is serviced third because it has the lowest bit value.

The interrupt on pin 2 is serviced last because it is the only remaining interrupt.

Table 19-1. Determining Interrupt Priority

Pin (bit)	3	2	1	0
Priority	High Level (Level A)	Low Level (Level B)	High Level (Level A)	Low Level (Level B)
Order	2	4	1	3

19.2.3.3 High-Level-Interrupt Block and Low-Level-Interrupt Block

The interrupt-handling blocks each contain two registers: an offset register and an emulation register. The high-level interrupts are denoted as level A and consequently, the registers for the high-level-interrupt-handling block are GIOOFFA (Section 19.6.7) and GIOEMUA (Section 19.6.9); see Figure 19-8. Likewise, the registers for the low-level-interrupt-handling block (denoted as level B) are GIOOFFB (Section 19.6.8) and GIOEMUB (Section 19.6.10); see Figure 19-9.

Figure 19-8. High-Level-Interrupt-Handling Block

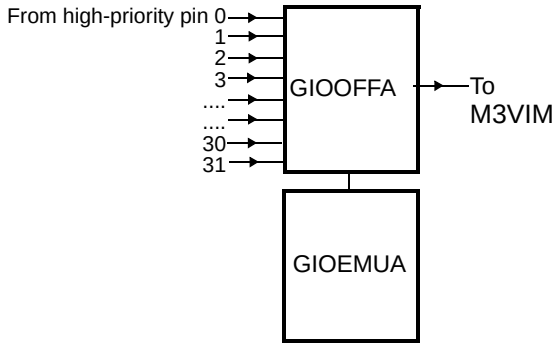
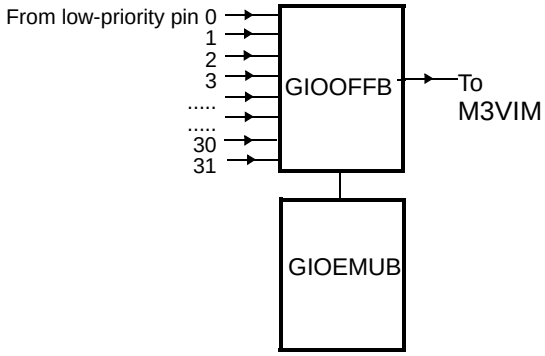


Figure 19-9. Low-Level-Interrupt-Handling Block



The read-only registers GIOOFFA and GIOOFFB generate a numerical offset value that represents the highest-priority pending external interrupt (see Table 19-2). The offset can be used to locate the position of the interrupt routine in the vector table. A read of the offset register clears the offset register and the corresponding flag bit in the GIOFLG register.

The high-level offset register, GIOOFFA, receives signals from each active interrupt that is configured as high-level. The GIOOFFA displays the high-level interrupt with the highest priority (that is, the interrupt that was generated by the lowest bit in the flag register). Similarly, the GIOOFFB displays the low-level interrupt with the highest priority.

The emulation control registers (GIOEMUA and GIOEMUB; Section 19.6.9 and Section 19.6.10) mirror the offset registers. The emulation registers contain a numerical offset value that represent the highest priority pending external interrupt (see Table 19-2). A read of the emulation registers does not clear any bit in any

register. These registers allow the device emulator to read and display the offset register values without affecting interrupt execution.

Table 19-2. GIO Offset A or B Values and Corresponding Interrupt

GIOOFF(5–0)	Pending External Interrupt
000000	No interrupt
000001	Interrupt 0
000010	Interrupt 1
000011	Interrupt 2
000100	Interrupt 3
. . .	. . .
100000	Interrupt 31
100001–111111	Reserved

Table 19-3 illustrates how to identify the interrupt being serviced. In this example, interrupts are enabled for pins 0, 1 and 2 and all three pins are set to the same interrupt level (same priority). This example assumes that interrupts on pins 0 and 2 occur simultaneously. Reading the flag control register, GIOFLG, returns a value of xxxxx101, indicating that interrupt flags have been set for pins 0 and 2. The first five values are indeterminate because only pins 0, 1 and 2 are interrupt-enabled. Reading the offset B control register, GIOOFFB, returns a value of 00000001, indicating that the interrupt on pin 0 is currently being processed (see Table 19-2).

Table 19-3. Reading the Offset Register to Determine Serviced Interrupt

Bits	7	6	5	4	3	2	1	0
GIOENA	Reserved					1	1	1
GIOPOL	Reserved					1	0	0
GIOFLG	Reserved					1	0	1
GIOLVL	Reserved					0	0	0
GIOOFFA	0	0	0	0	0	0	0	0
GIOEMUA	0	0	0	0	0	0	0	0
GIOOFFB	0	0	0	0	0	0	0	1
GIOEMUB	0	0	0	0	0	0	0	1

19.2.3.4 Special Considerations for Interrupts

Please note that interrupts are subject to the following special considerations:

- On devices where fewer than 32 interrupts are available, the unused control register bits are reserved.
- To use the enabled pins as interrupts, the I/O function of the pins are typically set as input. If the pin's I/O function is set as output, the signal feeds directly into the input. In this case, interrupts are only generated when the device toggles the data output register, thereby creating the appropriate interrupt edge. See Table 19.7.4.
- The interrupt flag can be set even though the interrupt is not enabled. Therefore, you must clear the flag register before enabling the interrupts to ensure that a spurious interrupt is not generated. See Section 19.7.4.
- If interrupts are enabled when GIODIN[7:0] is read, the bits corresponding to the enabled interrupts must be masked to avoid ambiguous results. For example, if GIO port A pins [2:0] are configured as interrupts and pins [7:3] are configured as inputs (V_{IH} applied), then a read of GIODIN0 will read 11111xxx, where the x values are interrupt levels and not inputs. Mask the input register against (in this case) 11111000. See section 19.7.1, *Example: Setting Interrupts and Configuring Pins for Output*, page 1093.

19.3 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)
  - Module level power-down
  - Device level power-down

19.3.1 Emulation Mode

Emulation mode is used by debugger tools to stop the CPU at breakpoints to read registers. When the device is in emulation mode, it pulls the suspend signal high.

Note: Emulation Mode
Emulation mode is a mode of operation of the device and is separate from the GIO emulation registers (GIOEMUA and GIOEMUB).

During emulation mode:

- External interrupts are not captured because the M3VIM is unable to service interrupts.
- Any register can be read without affecting the state of the system.
- A write to a register affects the state of the system.

19.3.2 Power-Down Mode (Low-Power Mode)

In the 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. The GIO module has two power-down modes: module-level power down and device-level power down. In both these power-down modes (low-power modes), interrupt pins become level-sensitive rather than edge-sensitive. The polarity bit changes function from falling edge and rising edge to low and high. A corresponding level on an interrupt pin pulls the module out of low-power mode.

19.3.2.1 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 Register for details.

19.3.2.2 Device-Level Power Down

The entire device can be placed in one of the pre-defined low-power modes: doze, snooze, sleep, or hibernate. See the device datasheet for details.

19.4 Pullup/Pulldown Function

GIO module pins can have either an active pullup or active pulldown that makes it possible to leave the pins unconnected externally when the pins are input pins. The pull capability is enabled by programming the GIOPULDIS[7:0] register (Section 19.6.17). By enabling the pull capability, the default pull on the GIO pins, as indicated in the device datasheet, are enabled. The pins also have the capability to be programmed as either a pullup or a pulldown using the GIOPSL[7:0] register (Section 19.6.18). Programming GIOPSL[7:0] overrides the default pull on the pins. See Table 19-4.

The default pullup/pulldown functionality of all GIO pins at system reset is fixed by design. Please see the TMS470M Series device-specific data sheet for the reset state of the pullups/pulldowns and for the current supplied by the pullups/pulldowns.

Note:
It is possible to disable the GIO Input buffer altogether by configuring the pin direction to input via the GIODIR[A-H][7:0], disabling the pin's pull functionality using GIOPDIS[7:0], and setting the pin's corresponding bit in the GIOPSEL[7:0] to 0. Once an input buffer is disabled, the external signal can not be read from the buffer internally. This note also applies to other peripheral pins (e.g HET pins) when they are used as a normal GIO pin.

Note:
The nTRST being high overrides the input buffer disable. This note also applies to other peripheral pins (e.g HET pins) when they are used as a normal GIO pin.

19.4.1 Summary

The behavior of the input buffer, output buffer, and the pull control is summarized in Table 19-4.

Table 19-4. Input Buffer, Output Buffer, and Pull Control Behavior as GPIO Pins

Device under Reset?	Pin Direction (DIR)	Pull Disable (PULDIS)	Pull Select (PSL)	Pull Control	Output Buffer	Input Buffer
Yes	X	X	X	Device- and module-specific	Disabled	Depends on pull control
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
2 DIR = 0 for input, 1 for output
3 PULDIS = 0 for enabling pull control
= 1 for disabling pull control
4 PSL= 0 for pull-down functionality
= 1 for pull-up functionality

19.5 Open Drain Function

The GIO pins can be configured to include an open drain functionality using the GIOPDR[7:0] registers (Section 19.6.16), when the pins are configured as output pins. When the open drain functionality is enabled (GIOPDR[7:0] = 1), a 0 written to the data output register GIODOUT[7:0] (Section 19.6.13) forces the pin to a low output voltage (V_{OL} or lower), whereas a 1 written to the data output register forces the pin to a high impedance state. The open drain functionality is disabled when the pin is configured as an input pin.

19.6 GIO Control Registers

Table 19-5 shows the summary of the GIO registers. It assumes eight ports designated A–H (four of them interrupt capable ports), with eight pins per port. As an example, port B bits [2:0] of the GIODIR register would be represented as GIODIR[B][2:0]. Each port has a set of registers that control the I/O function of the pins for that port - they are GIODIR[A-H][7:0], GIODIN[A-H][7:0], GIODOUT[A-H][7:0], GIODSET[A-H][7:0], GIODCLR[A-H][7:0], GIOPDR[A-H][7:0], GIOPULDIS[A-H][7:0], and GIOPSL[A-H][7:0].

The registers are accessible in 8-, 16-, and 32-bit reads or writes. Consult the device-specific data sheet to verify the pin configuration. The start address for the GIO module is 0xFFF7 BC00.

Table 19-5. GIO Control Register Summary

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
0x00 GIOGCR0 Page 1066	Reserved															
	Reserved															RESET
0x08 GIOINTDET Page 1067	GIOINTDET 3[7:0]								GIOINTDET 2[7:0]							
	GIOINTDET 1[7:0]								GIOINTDET 0[7:0]							
0x0C GIOPOL Page 1069	GIOPOL 3[7:0]								GIOPOL 2[7:0]							
	GIOPOL 1[7:0]								GIOPOL 0[7:0]							
0x10 GIOENASET Page 1071	GIOENASET 3[7:0]								GIOENASET 2[7:0]							
	GIOENASET 1[7:0]								GIOENASET 0[7:0]							
0x14 GIOENACLR Page 1071	GIOENACLR 3[7:0]								GIOENACLR 2[7:0]							
	GIOENACLR 1[7:0]								GIOENACLR 0[7:0]							
0x18 GIOLVL SET Page 1074	GIOLVLSET 3[7:0]								GIOLVLSET 2[7:0]							
	GIOLVLSET 1[7:0]								GIOLVLSET 0[7:0]							
0x1C GIOLVLCLR Page 1074	GIOLVLCLR 3[7:0]								GIOLVLCLR 2[7:0]							
	GIOLVLCLR 1[7:0]								GIOENACLR 0[7:0]							

1 See the specific device data sheet to verify the base address of the GIO registers.

Table 19-5. GIO Control Register Summary (Continued)

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
0x20 GIOFLG Page 1078	GIOFLG 3[7:0]								GIOFLG 2[7:0]							
	GIOFLG 1[7:0]								GIOFLG 0[7:0]							
0x24 GIOOFFA Page 1080	Reserved															
	Reserved										GIOOFFA[5:0]					
0x28 GIOOFFB Page 1081	Reserved															
	Reserved										GIOOFFB[5:0]					
0x2C GIOEMUA Page 1082	Reserved															
	Reserved										GIOEMUA[5:0]					
0x30 GIOEMUB Page 1083	Reserved															
	Reserved										GIOEMUB[5:0]					
0x34, 54, 74, 94, B4, D4, F4, 114 GIODIR[7:0] Page 1084	Reserved															
	Reserved								GIODIR0[A-H][7:0]							
0x38, 58, 78, 98, B8, D8, F8, 118 GIODIN[7:0] Page 1085	Reserved															
	Reserved								GIODIN0[A-H][7:0]							
0x3C, 5C, 7C, 9C, BC, DC, FC, 11C GIODOUT[7:0] Page 1086	Reserved															
	Reserved								GIODOUT0[A-H][7:0]							
0x40, 60, 80, A0, C0, E0, 100, 120 GIOSET[7:0] Page 1087	Reserved															
	Reserved								GIOSET[A-H][7:0]							

1 See the specific device data sheet to verify the base address of the GIO registers.

Table 19-5. GIO Control Register Summary (Continued)

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
0x44, 64, 84, A4, C4, E4, 104, 124 GIOCLR[7:0] Page 1088	Reserved															
	Reserved								GIOCLR[A-H][7:0]							
0x48, 68, 88, A8, C8, E8, 108, 128 GIOPDR[7:0] Page 1089	Reserved															
	Reserved								GIOPDR[A-H][7:0]							
0x4C, 6C, 8C, AC, CC, EC, 10C, 12C GIOPULDIS[7:0] Page 1090	Reserved															
	Reserved								GIOPULDIS[A-H][7:0]							
0x50, 70, 90, B0, D0, F0, 110, 130 GIOPSL[7:0] Page 1091	Reserved															
	Reserved								GIOPSL[A-H][7:0]							

1 See the specific device data sheet to verify the base address of the GIO registers.

19.6.1 GIO Global Control Register (GIOGCR0)

The GIOGCR0 register contains one bit that controls the module reset status. Writing a zero (0) to this bit puts the module in a reset state. After system reset, this bit must be set to 1 before normal operations can begin on this module. Figure 19-10 and Table 19-6 describe this register.

Figure 19-10. GIO Global Control Register (GIOGCR0) [offset = 0x00]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															RESET
R-0															R/WP-0

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 19-6. GIO Global Control Register (GIOGCR0) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved		Reads return zeros and writes have no effect.
0	RESET		GIO reset.
		0	The GIO is in reset state.
		1	The GIO is operating normally.

19.6.2 GIO Interrupt Detect Register (GIOINTDET)

The GIOINTDET register provides the flexibility to either ignore the polarity of the edges that are recognized as an interrupt, in which case both rising and falling edges are recognized, or recognizing the interrupt on specifically a rising or falling edge as determined by the GIOPOL register (Section 19.6.3). To ensure recognition of the signal as an edge, the signal must maintain the new level for at least one VCLK cycle. Figure 19-11 and Table 19-7 describe this register.

Figure 19-11. GIO Interrupt Detect Register (GIOINTDET) [offset = 0x08]

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

R = Read in all modes; W = Write in user and privileged modes; -n = Value after reset

.

Table 19-7. GIO Interrupt Detect Register (GIOINTDET) Field Descriptions

Bit	Name	Value	Description
31–24	GIOINTDET 3[7:0]		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; Section 19.6.3).
		1	The flag sets on both the rising and falling edges on the corresponding pin.
23–16	GIOINTDET 2[7:0]		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; Section 19.6.3).
		1	The flag sets on both the rising and falling edges on the corresponding pin.
15–8	GIOINTDET 1[7:0]		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; Section 19.6.3).
		1	The flag sets on both the rising and falling edges on the corresponding pin.

Table 19-7. GIO Interrupt Detect Register (GIOINTDET) Field Descriptions (Continued)

Bit	Name	Value	Description
7–0	GIOINTDET 0[7:0]		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; Section 19.6.3).
		1	The flag sets on both the rising and falling edges on the corresponding pin.

19.6.3 GIO Interrupt Polarity Register (GIOPOL)

The GIOPOL register controls the polarity—rising edge (low to high) or falling edge (high to low)—that sets the flag. To ensure recognition of the signal as an edge, the signal must maintain the new level for at least one VCLK cycle. When the device is in low power mode, the interrupts are no longer triggered by an edge, but instead by a level. Therefore, in low power mode, the GIOPOL register controls the ***level***, high or low, which will trigger the interrupt. Figure 19-12 and Table 19-8 describe this register.

Figure 19-12. GIO Interrupt Polarity Register (GIOPOL) [offset = 0x0C]

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

R = Read in all modes; W = Write in user and privileged modes; -n = Value after reset

.

Table 19-8. GIO Interrupt Polarity Register (GIOPOL) Field Descriptions

Bit	Name	Value	Description
31–24	GIOPOL 3[7:0]		Interrupt polarity select for pins GIOD[7:0].
			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 (doze, snooze, sleep or hibernate):
		0	The interrupt is triggered on the low level.
		1	The interrupt is triggered on the high level.
23–16	GIOPOL 2[7:0]		Interrupt polarity select for pins GIOC[7:0].
			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 (doze, snooze, sleep or hibernate):
		0	The interrupt is triggered on the low level.
		1	The interrupt is triggered on the high level.
15–8	GIOPOL 1[7:0]		Interrupt polarity select for pins GIOB[7:0].
			User or privileged mode:
		0	The flag is set on the falling edge on the corresponding pin.

Table 19-8. GIO Interrupt Polarity Register (GIOPOL) Field Descriptions (Continued)

Bit	Name	Value	Description
		1	The flag is set on the rising edge on the corresponding pin.
			Low-power mode (doze, snooze, sleep or hibernate):
		0	The interrupt is triggered on the low level.
		1	The interrupt is triggered on the high level.
7–0	GIOPOL 0[7:0]		Interrupt polarity select for pins GIOA[7:0].
			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 (doze, snooze, sleep or hibernate):
		0	The interrupt is triggered on the low level.
		1	The interrupt is triggered on the high level.

19.6.4 GIO Interrupt Enable Registers (GIOENASET and GIOENACLR)

The GIOENASET and GIOENACLR register controls which interrupt-capable pins are configured as interrupts. These two registers are physically implemented as a single register. If the interrupt is enabled, the signal with an appropriate edge leads to an interrupt.

19.6.4.1 GIOENASET Register

Figure 19-13 and Table 19-9 describe this register.

Note: Enabling Interrupt at the Device Level
Two bits, corresponding to the GIO high level (level A) and low level (level B) interrupts, must be set within the vectored interrupt manager (M3VIM) in the interrupt mask register (REQMASK;) to enable the appropriate interrupts. Additionally, the ARM CPU must be configured to recognize interrupt requests (IRQ/FIQ); .

Figure 19-13. GIO Interrupt Enable Register (GIOENASET) [offset = 0x10]

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

R = Read in all modes; W = Write in user and privileged modes; -n = Value after reset

.

Table 19-9. GIO Interrupt Enable Register (GIOENASET) Field Descriptions

Bit	Name	Value	Description
31–24	GIOENASET 3[7:0]		Interrupt enable for pins GIOD[7:0]
		0	*Read:* The interrupt is disabled. *Write:* Writing a zero to this bit has no effect.
		1	*Read or write:* The interrupt is enabled.
23–16	GIOENASET 2[7:0]		Interrupt enable for pins GIOC[7:0]
		0	*Read:* The interrupt is disabled. *Write:* Writing a zero to this bit has no effect.
		1	*Read or write:* The interrupt is enabled.
15–8	GIOENASET 1[7:0]		Interrupt enable for pins GIOB[7:0]
		0	*Read:* The interrupt is disabled. *Write:* Writing a zero to this bit has no effect.
		1	*Read or write:* The interrupt is enabled.
7–0	GIOENASET 0[7:0]		Interrupt enable for pins GIOA[7:0]

Table 19-9. GIO Interrupt Enable Register (GIOENASET) Field Descriptions (Continued)

Bit	Name	Value	Description
		0	*Read:* The interrupt is disabled. *Write:* Writing a zero to this bit has no effect.
		1	*Read or write:* The interrupt is enabled.

19.6.4.2 GIOENACLR Register

This register disables the interrupt. Figure 19-14 and Table 19-10 describe this register.

Figure 19-14. GIO Interrupt Enable Register (GIOENACLR) [offset = 0x14]

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

R = Read in all modes; W = Write in user and privileged modes; -n = Value after reset

.

Table 19-10. GIO Interrupt Enable Register (GIOENACLR) Field Descriptions

Bit	Name	Value	Description
31–24	GIOENACLR 3[7:0]		Interrupt disable for pins GIOD[7:0].
		0	*Read:* The interrupt is disabled. *Write:* Writing a zero to this bit has no effect.
		1	*Read:* The interrupt is enabled. *Write:* Writing a one to this bit disables the interrupt.
23–16	GIOENACLR 2[7:0]		Interrupt disable for pins GIOC[7:0].
		0	*Read:* The interrupt is disabled. *Write:* Writing a zero to this bit has no effect.
		1	*Read:* The interrupt is enabled. *Write:* Writing a one to this bit disables the interrupt.
15–8	GIOENACLR 1[7:0]		Interrupt disable for pins GIOB[7:0].
		0	*Read:* The interrupt is disabled. *Write:* Writing a zero to this bit has no effect.
		1	*Read:* The interrupt is enabled. *Write:* Writing a one to this bit disables the interrupt.
7–0	GIOENACLR 0[7:0]		Interrupt disable for pins GIOA[7:0].
		0	*Read:* The interrupt is disabled. *Write:* Writing a zero to this bit has no effect.
		1	*Read:* The interrupt is enabled. *Write:* Writing a one to this bit disables the interrupt.

19.6.5 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 M3VIM. Each interrupt is individually configured.

- The high-level interrupts are recorded to GIOOFFA and GIOEMUA.
- The low-level interrupts are recorded to GIOOFFB and GIOEMUB.

19.6.5.1 GIOLVLSET Register

The GIOLVLSET register is used to configure an interrupt as a high-level interrupt going to the M3VIM. An interrupt can be configured as a high level interrupt by writing a 1 into the corresponding bit of the GIOLVLSET register. Writing a zero has no effect. Figure 19-15 and Table 19-11 describe this register.

Figure 19-15. GIO Interrupt Priority Register (GIOLVSLSET) [offset = 0x18]

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

R = Read in all modes; W = Write in user and privileged modes; -n = Value after reset

.

Table 19-11. GIO Interrupt Priority Register (GIOLVSLSET) Field Descriptions

Bit	Name	Value	Description
31-24	GIOLVSLSET 3[7:0]		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 GIOOFFB and GIOEMUB. *Write:* Writing a zero to this bit has no effect.
		1	*Read or write:* The interrupt is set as a high level interrupt. The high-level interrupts are recorded to GIOOFFA and GIOEMUA.
23-16	GIOLVSLSET 2[7:0]		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 GIOOFFB and GIOEMUB. *Write:* Writing a zero to this bit has no effect.
		1	*Read or write:* The interrupt is set as a high level interrupt. The high-level interrupts are recorded to GIOOFFA and GIOEMUA.
15-8	GIOLVSLSET 1[7:0]		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 GIOOFFB and GIOEMUB. *Write:* Writing a zero to this bit has no effect.
		1	*Read or write:* The interrupt is set as a high level interrupt. The high-level interrupts are recorded to GIOOFFA and GIOEMUA.

Table 19-11. GIO Interrupt Priority Register (GIOLVSLSET) Field Descriptions (Continued)

Bit	Name	Value	Description
7-0	GIOLVSLSET 0[7:0]		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 GIOOFFB and GIOEMUB. *Write:* Writing a zero to this bit has no effect.
		1	*Read or write:* The interrupt is set as a high level interrupt. The high-level interrupts are recorded to GIOOFFA and GIOEMUA.

19.6.5.2 GIOLVLCLR Register

The GIOLVLCLR register is used to configure an interrupt as a low level interrupt going to the M3VIM. Figure 19-16 and Table 19-12 describe this register.

Figure 19-16. GIO Interrupt Priority Register (GIOLVLCLR) [offset = 0x1C]

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

R = Read in all modes; W = Write in user and privileged modes; -n = Value after reset

.

Table 19-12. GIO Interrupt Priority Register (GIOLVLCLR) Field Descriptions

Bit	Name	Value	Description
31–24	GIOLVLCLR 3[7:0]		GIO low priority interrupt for pins GIOD[7:0].
		0	*Read:* The interrupt is a low-level interrupt. *Write:* Writing a zero to this bit has no effect.
		1	*Read:* The interrupt is set as a high level interrupt. The high-level interrupts are recorded to GIOOFFA and GIOEMUA. *Write:* The interrupt is set as a low level interrupt. The low-level interrupts are recorded to GIOOFFB and GIOEMUB.
23–16	GIOLVLCLR 2[7:0]		GIO low priority interrupt for pins GIOC[7:0].
		0	*Read:* The interrupt is a low-level interrupt. *Write:* Writing a zero to this bit has no effect.
		1	*Read:* The interrupt is set as a high level interrupt. The high-level interrupts are recorded to GIOOFFA and GIOEMUA. *Write:* The interrupt is set as a low level interrupt. The low-level interrupts are recorded to GIOOFFB and GIOEMUB.
15–8	GIOLVLCLR 1[7:0]		GIO low priority interrupt for pins GIOB[7:0].
		0	*Read:* The interrupt is a low-level interrupt. *Write:* Writing a zero to this bit has no effect.
		1	*Read:* The interrupt is set as a high level interrupt. The high-level interrupts are recorded to GIOOFFA and GIOEMUA. *Write:* The interrupt is set as a low level interrupt. The low-level interrupts are recorded to GIOOFFB and GIOEMUB.

Table 19-12. GIO Interrupt Priority Register (GIOLVLCLR) Field Descriptions (Continued)

Bit	Name	Value	Description
7–0	GIOLVLCLR 0[7:0]		GIO low priority interrupt for pins GIOA[7:0].
		0	*Read:* The interrupt is a low-level interrupt. *Write:* Writing a zero to this bit has no effect.
		1	*Read:* The interrupt is set as a high level interrupt. The high-level interrupts are recorded to GIOOFFA and GIOEMUA. *Write:* The interrupt is set as a low level interrupt. The low-level interrupts are recorded to GIOOFFB and GIOEMUB.

19.6.6 GIO Interrupt Flag Register (GIOFLG)

The GIOFLG register contains flags indicating that the transition edge (as set in GIOINTDET;Section 19.6.2 and GIOPOL; Section 19.6.3) has occurred. The flag is also cleared by reading the appropriate offset register; see Section 19.2.3.3. Figure 19-17 and Table 19-13 describe this register.

Figure 19-17. GIO Interrupt Flag Register (GIOFLG) [offset = 0x20]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GIOFLG 3[7:0]								GIOFLG 2[7:0]							
R/WC-0								R/WC-0							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
GIOFLG 1[7:0]								GIOFLG 0[7:0]							
R/WC-0								R/WC-0							

R = Read in all modes; WC = Write clears the bit, -n = Value after reset

.

Table 19-13. GIO Interrupt Flag Register (GIOFLG) Field Descriptions

Bit	Name	Value	Description
31–24	GIOFLG 3[7:0]		GIO flag for pins GIOD[7:0].
		0	*Read:* A transition has not occurred since the last clear. *Write:* Writing a zero to this bit has no effect.
		1	*Read:* A transition has occurred since the last clear. *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[7:0]		GIO flag for pins GIOC[7:0].
		0	*Read:* A transition has not occurred since the last clear. *Write:* Writing a zero to this bit has no effect.
		1	*Read:* A transition has occurred since the last clear. *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[7:0]		GIO flag for pins GIOB[7:0].
		0	*Read:* A transition has not occurred since the last clear. *Write:* Writing a zero to this bit has no effect.
		1	*Read:* A transition has occurred since the last clear. *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[7:0]		GIO flag for pins GIOA[7:0].

Table 19-13. GIO Interrupt Flag Register (GIOFLG) Field Descriptions (Continued)

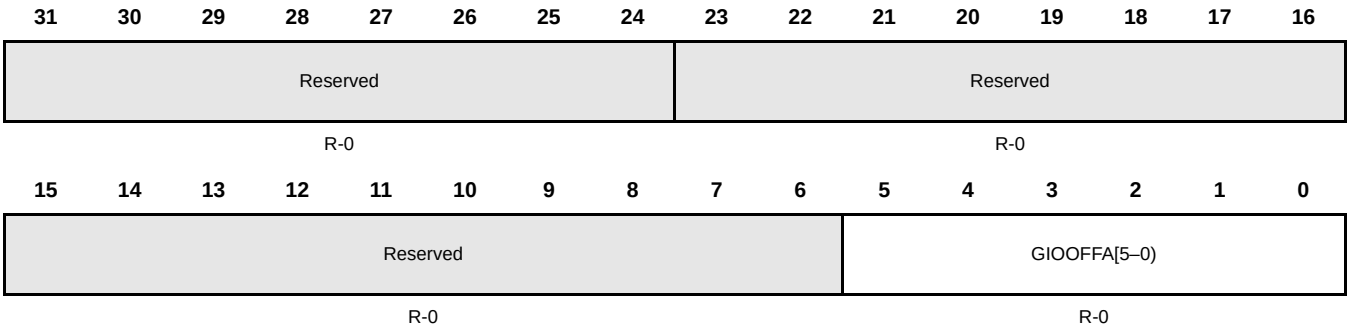
Bit	Name	Value	Description
		0	*Read:* A transition has not occurred since the last clear. *Write:* Writing a zero to this bit has no effect.
		1	*Read:* The selected transition on the corresponding pin has occurred. *Write:* The corresponding bit is cleared to 0. **Note: This bit is also cleared by a read to the corresponding bit in the appropriate offset register**

19.6.7 GIO Offset A Register (GIOOFFA)

The GIOOFFA 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 19-18 and Table 19-14 describe this register.

Note:
Reading this register clears it and the corresponding flag bit in the GIOFLG register; Section 19.6.6. However, in emulation mode, a read to this register does not clear the corresponding flag bit.

Figure 19-18. GIO Offset A Register (GIOOFFA) [offset = 0x24]



R = Read in all modes; W = Write in user and privileged modes; -n = Value after reset

.

Table 19-14. GIO Offset A Register (GIOOFFA) Field Descriptions

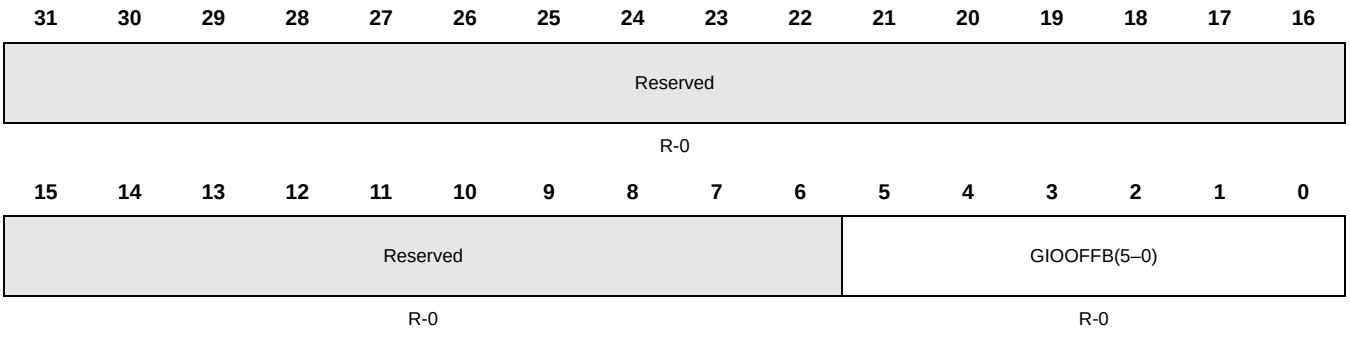
Bit	Name	Value	Description
31–6	Reserved		Reads return zero and writes have no effect.
5–0	GIOOFFA(5–0)		GIO offset A. These bits index the currently pending high-priority interrupt. This register and the flag bit (in the GIOFLG register) are also cleared when this register is read, except in emulation mode.
		000000	No interrupt is pending.
		000001	Interrupt 0 is pending with a high priority.
		. . .	. . .
		100000	Interrupt 31 is pending with a high priority.
		100001–111111	Reserved

19.6.8 GIO Offset B Register (GIOOFFB)

The GIOOFFB 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 19-19 and Table 19-15 describe this register.

Note:
Reading this register clears it and the corresponding flag bit in the GIOFLG register; Section 19.6.6. However, in emulation mode, a read to this register does not clear the corresponding flag bit.

Figure 19-19. GIO Offset B Register (GIOOFFB) [offset = 0x28]



R = Read in all modes; W = Write in user and privileged modes; -n = Value after reset

Table 19-15. GIO Offset B Register (GIOOFFB) Field Descriptions

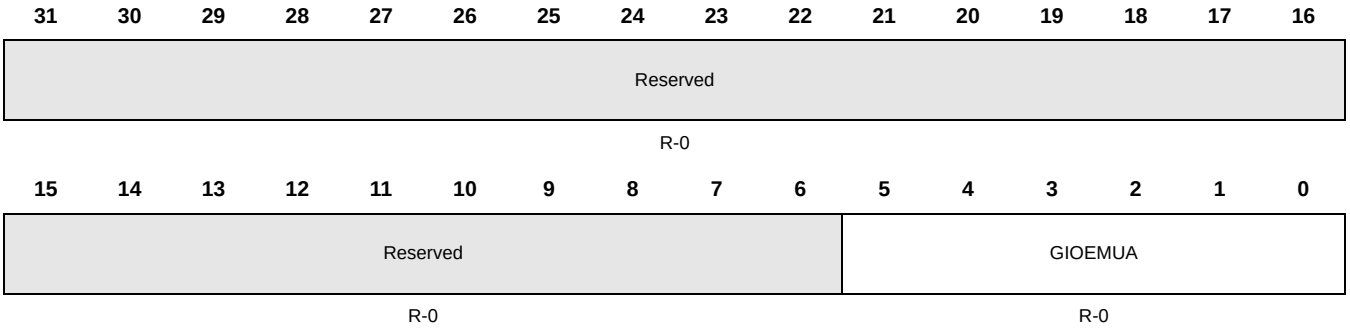
Bit	Name	Value	Description
31–6	Reserved		Reads return zero and writes have no effect.
5–0	GIOOFFB		GIO offset register B. These bits index the currently pending low-priority interrupt. This register and the flag bit (in the GIOFLG register) are also cleared when this register is read, except in emulation mode.
		000000	No interrupt is pending.
		000001	Interrupt 0 is pending with a low priority.
		. . .	. . .
		100000	Interrupt 31 is pending is pending with a low priority.
		100001–111111	Reserved

19.6.9 GIO Emulation A Register (GIOEMUA)

The GIOEMUA register is a read-only register, and is provided for use by the debug monitor in normal operation. The contents of this register are identical to the contents of GIOOFFA. Figure 19-20 and Table 19-16 describe this register.

Note:
The corresponding flag in the GIOFLG register (Section 19.6.6) is not cleared when the GIOEMUA register is read.

Figure 19-20. GIO Emulation A Register (GIOEMUA) [offset = 0x2C]



R = Read in all modes; n = Value after reset

Table 19-16. GIO Emulation A Register (GIOEMUA) Field Descriptions

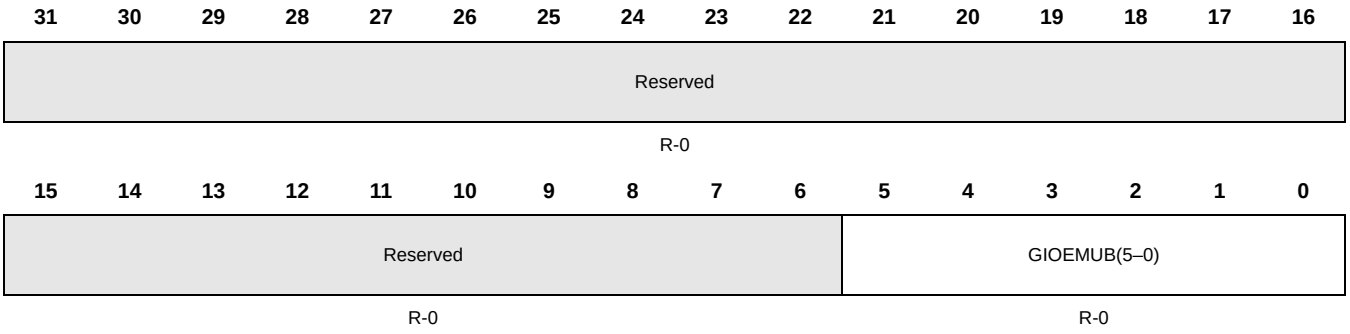
Bit	Name	Value	Description
31–6	Reserved		Reads return zero and writes have no effect.
5–0	GIOEMUA		GIO emulation register A. These bits index the currently pending high-priority interrupt.
		000000	No interrupt is pending.
		000001	Interrupt 0 is pending with a high priority.
		. . .	. . .
		100000	Interrupt 31 is pending with a high priority.
		100001–111111	Reserved

19.6.10 GIO Emulation B Register (GIOEMUB)

The GIOEMUB register is a read-only register, and is provided for use by the debug monitor in normal operation. The contents of this register are identical to the contents of GIOOFFB. Figure 19-21 and Table 19-17 describe this register.

Note:
The corresponding flag in the GIOFLG register (Section 19.6.6) is not cleared when the GIOEMUB register is read.

Figure 19-21. GIO Emulation B Register (GIOEMUB) [offset = 0x30]



R = Read in all modes; -n = Value after reset

.

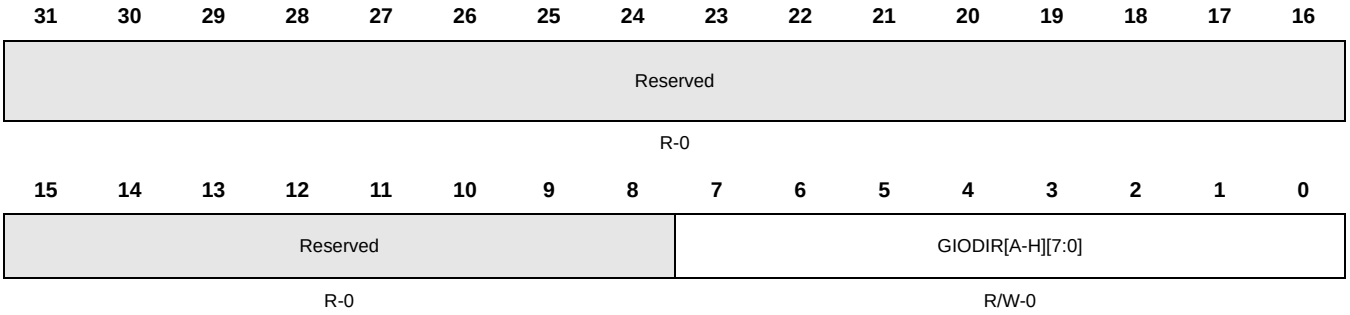
Table 19-17. GIO Emulation B Register (GIOEMUB) Field Descriptions

Bit	Name	Value	Description
31–6	Reserved		Reads return zero and writes have no effect.
5–0	GIOEMUB		GIO emulation register B. These bits index the currently pending low-priority interrupt.
		000000	No interrupt is pending.
		000001	Interrupt 0 is pending with a low priority.
		. . .	. . .
		100000	Interrupt 31 is pending with a low priority.
		100001–111111	Reserved

19.6.11 GIO Data Direction Registers [A-H][7:0] (GIODIR[A-H][7:0])

The GIODIR register controls whether the pins of a given port are configured as inputs or outputs. Figure 19-22 and Table 19-18 describe this register.

Figure 19-22. GIO Data Direction Registers [A-H][7:0] (GIODIR[A-H][7:0])



R = Read in all modes; W = Write in user and privileged modes; -n = Value after reset

.

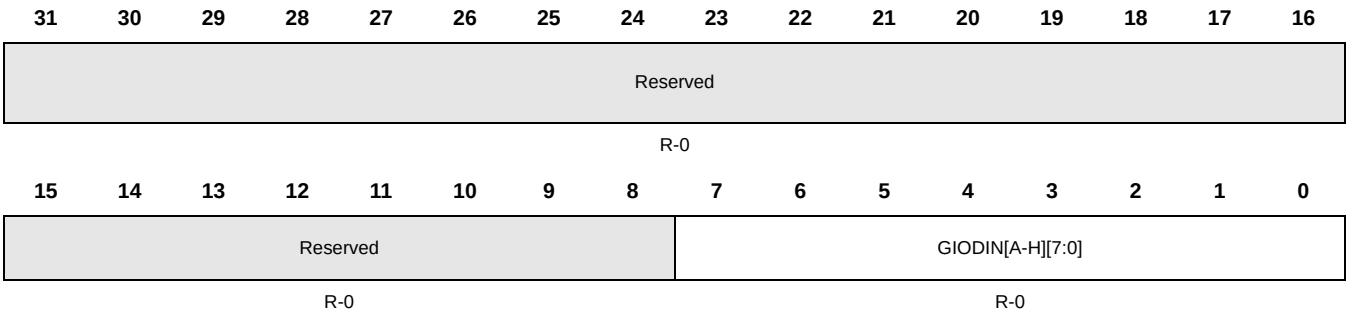
Table 19-18. GIO Data Direction Registers [A-H][7:0] (GIODIR[A-H][7:0]) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return zero and writes have no effect.
7–0	GIODIR[A-H][7:0]		GIO data direction of ports [A-H], pins[7:0].
		0	The output buffer is disabled.
		1	The output buffer is enabled.

19.6.12 GIO Data Input Registers [A-H][7:0] (GIODIN[A-H][7:0])

Values in the GIODIN register reflect the current state (high = 1 or low = 0) on the pins of the port. Figure 19-23 and Table 19-19 describe this register.

Figure 19-23. GIO Data Input Registers [A-H][7:0] (GIODIN[A-H][7:0])



R = Read in all modes; W = Write in user and privileged modes; -n = Value after reset

Table 19-19. GIO Data Input Registers [A-H][7:0] (GIODIN[A-H][7:0]) Field Descriptions

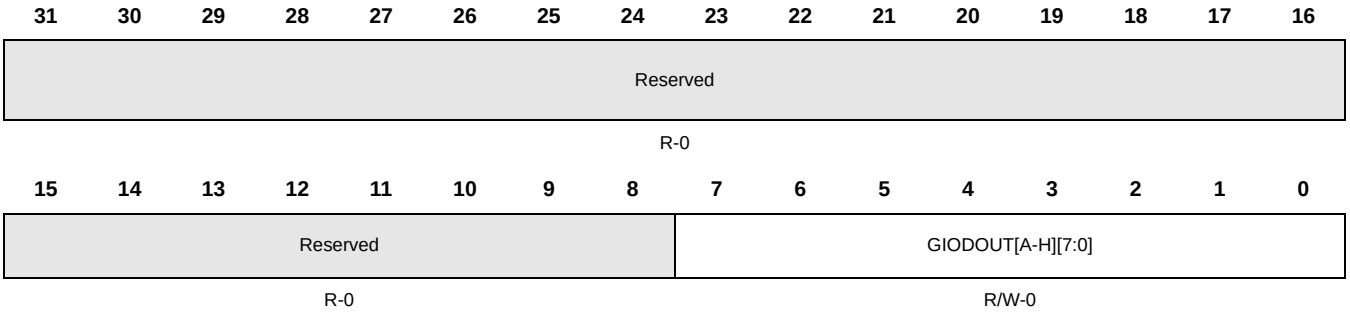
Bit	Name	Value	Description
31–8	Reserved		Reads return zero and writes have no effect.
7-0	GIODIN[A-H][7:0]		GIO data input for ports [H-A], pins[7:0].
		0	The pin is at logic low (0); input voltage is V_{IL} or lower.
		1	The pin is at logic high (1); input voltage is V_{IH} or higher.

19.6.13 GIO Data Output Registers [A-H][7:0] (GIODOUT[A-H][7:0])

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 19-24 and Table 19-20 describe this register.

Note:
Values in the GIODSET register, Section 19.6.14, set the data output control register bits to 1 regardless of the current value in the GIODOUT bits.

Figure 19-24. GIO Data Output Registers [A-H][7:0] (GIODOUT[A-H][7:0])



R = Read in all modes; W = Write in user and privileged modes; -n = Value after reset

.

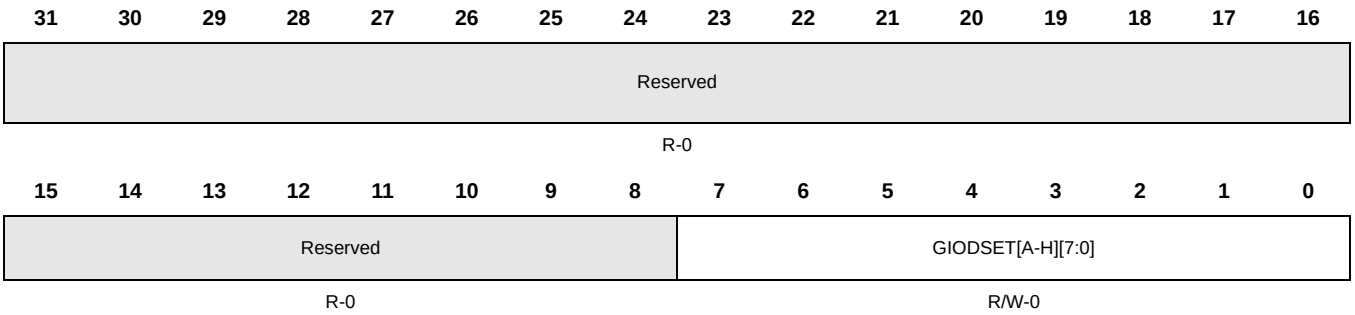
Table 19-20. GIO Data Output Registers [A-H][7:0] (GIODOUT[A-H][7:0]) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return zero and writes have no effect.
7-0	GIODOUT[A-H][7:0]		GIO data output of ports [A-H], pins[7:0].
		0	The pin is at logic low (0); output voltage is V_{OL} or lower.
		1	The pin is at logic high (1) (output voltage is V_{OH} or higher if the GIOPDRx bit = 0 and output is in high impedance state if the GIOPDRx bit = 1)

19.6.14 GIO Data Set Register [A-H][7:0] (GIODSET[A-H][7:0])

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 19-25 and Table 19-21 describe this register.

Figure 19-25. GIO Data Set Registers [A-H][7:0] (GIODSET[A-H][7:0])



R = Read in all modes; W = Write in user and privileged modes; -n = Value after reset

.

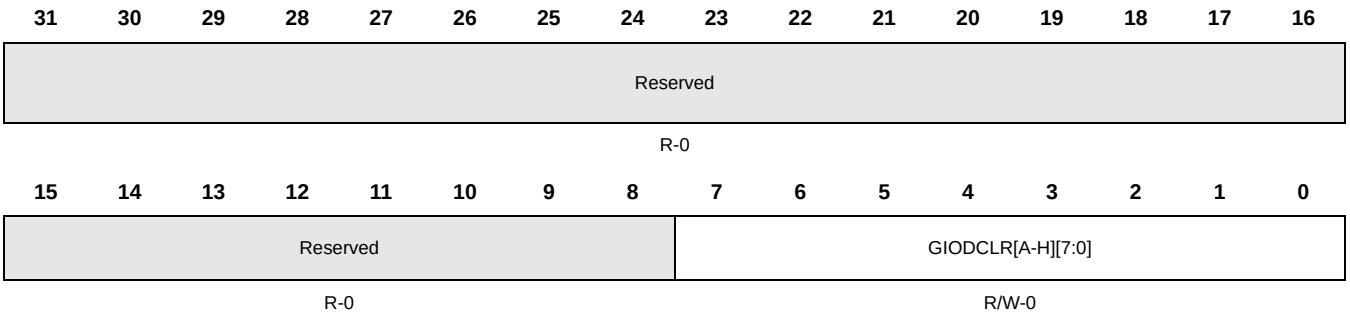
Table 19-21. GIO Data Set Registers [A-H][7:0] (GIODSET[A-H][7:0]) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return zero and writes have no effect.
7-0	GIODSET[A-H][7:0]		GIO data set for ports [A-H], pins[7:0].
		0	*Read:* The pin is at logic low (0); output voltage is V_{OL} or lower. *Write:* Writing a zero to this bit has no effect.
		1	*Read:* The pin is at logic high (1); output voltage is V_{OH} or higher. *Write:* The corresponding bit in GIODOUT[7:0] is set to 1.

19.6.15 GIO Data Clear Registers [A-H][7:0] (GIODCLR[A-H][7:0])

Values in this register clear the data output register (Section 19.6.14) bit to 0 regardless of its current value. The contents of this register reflect the contents of GIODOUT. Figure 19-26 and Table 19-22 describe this register.

Figure 19-26. GIO Data Clear Registers [A-H][7:0] (GIODCLR[A-H][7:0])



R = Read in all modes; W = Write in user and privileged modes; -n = Value after reset

Table 19-22. GIO Data Clear Registers [A-H][7:0] (GIODCLR[A-H][7:0]) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return zero and writes have no effect.
7-0	GIODCLR[A-H][7:0]		GIO data clear for ports [A-H], pins[7:0].
		0	*Read:* The pin is at logic low (0); output voltage is V_{OL} or lower. *Write:* Writing a zero to this bit has no effect on the corresponding bit in GIODOUT[7:0].
		1	*Read:* The pin is at logic high (1); output voltage is V_{OH} or higher. *Write:* The corresponding bit in GIODOUT[7:0] is cleared to 0.

19.6.16 GIO Open Drain Register [A-H][7:0] (GIOPDR[A-H][7:0])

Values in this register enable or disable the open drain capability of the data pins. Figure 19-27 and Table 19-23 describe this register.

Figure 19-27. GIO Open Drain Registers [A-H][7:0] (GIOPDR[A-H][7:0])

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								GIOPDR[A-H][7:0]							
R-0								R/W-0							

R = Read in all modes; W = Write in user and privilege modes; -n = Value after reset

.

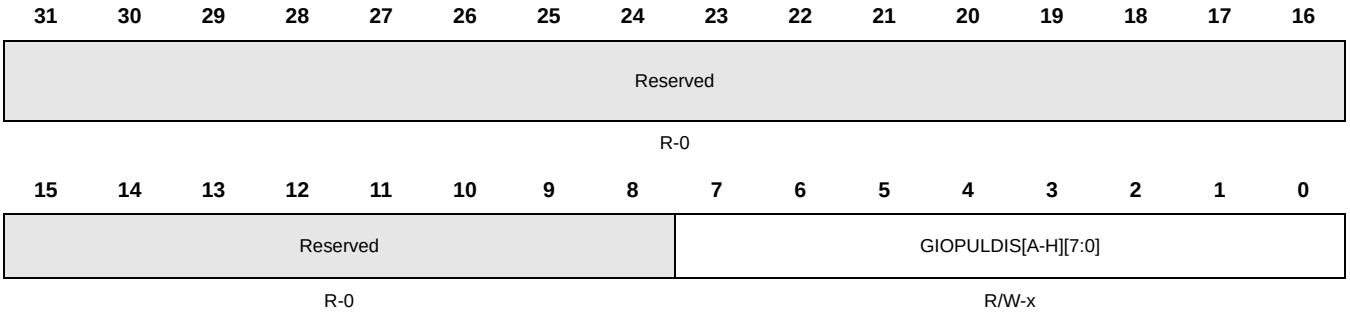
Table 19-23. GIO Open Drain Registers [A-H][7:0] (GIOPDR[A-H][7:0]) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return zero and writes have no effect.
7-0	GIOPDR[A-H][7:0]		GIO open drain for ports [A-H], pins[7:0].
		0	The pin is at logic low (0). Open drain functionality is disabled; the output voltage is V_{OL} or lower if GIODOUT =0 and V_{OH} or higher if GIODOUT =1.
		1	The pin is at logic high (1). Open drain functionality is enabled; the output voltage is V_{OL} or lower if GIODOUT =0 and z if GIODOUT =1.

19.6.17 GIO Pull Disable Registers [A-H][7:0] (GIOPULDIS[A-H][7:0])

Values in this register enable or disable the pull control capability of the pins. Figure 19-28 and Table 19-24 describe this register.

Figure 19-28. GIO Pull Disable Registers [A-H][7:0] (GIOPULDIS[A-H][7:0])



R = Read in all modes; W = Write in user and privileged modes; -n = Value after reset depends upon the device

.

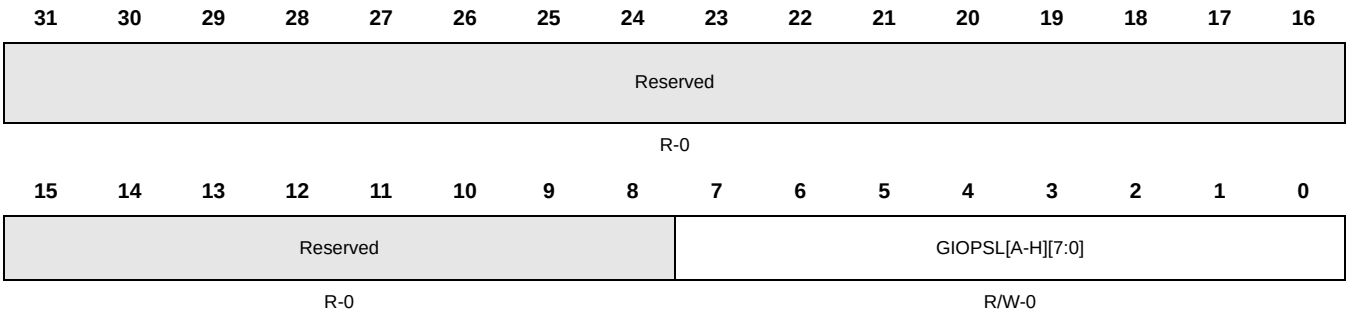
Table 19-24. GIO Pull Disable Registers [A-H][7:0] (GIOPULDIS[A-H][7:0]) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return zero and writes have no effect.
7–0	GIOPULDIS[A-H][7:0]		GIO pull disable for ports [A-H], pins[7:0]. Please refer to the device-specific datasheet for the reset state of these bits.
		0	The pull functionality is enabled.
		1	The pull functionality is disabled.

19.6.18 GIO Pull Select Register [A-H][7:0] (GIOPSL[A-H][7:0])

Values in this register select the pull up or pull down functionality of the pins. Figure 19-29 and Table 19-25 describe this register.

Figure 19-29. GIO Pull Select Registers [A-H][7:0] (GIOPSL[A-H][7:0])



R = Read in all modes; W = Write in user and privileged modes; -n = Value after reset

Table 19-25. GIO Pull Select Registers [A-H][7:0] (GIOPSL[A-H][7:0]) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return zero and writes have no effect.
7–0	GIOPSL[A-H][7:0]		GIO pull select for ports [A-H], pins[7:0].
		0	The pin is at logic low (0); the pull down functionality is enabled.
		1	The pin is at logic high (1); the pull up functionality is enabled.

19.7 Applications

The application examples in this section assume a typical configuration of five I/O functional ports, in which only Port A has three external interrupt-capable pins, 2:0. Table 19-26 illustrates the sample GIO register.

Table 19-26. Example GIO Register Set Showing Reserved Bits

Register	Address	7	6	5	4	3	2	1	0
GIOGCR	0x00								
GIOINTDET	0x08								
GIOPOL	0x0C								
GIOENASET	0x10								
GIOENACLR	0x14								
GIOLVLSET	0x18								
GIOLVLCLR	0x1C								
GIOFLG	0x20								
GIOOFFA	0x24								
GIOOFFB	0x28								
GIOEMUA	0x2C								
GIOEMUB	0x30								
GIODIRA	0x34								
GIODINA	0x38								
GIODOUTA	0x3C								
GIODSETA	0x40								
GIODCLRA	0x44								
GIOPDRA	0x48								
GIOPULDISA	0x4C								
GIOPSLA	0x50								
GIODIRB	0x54								
GIODINB	0x58								
GIODOUTB	0x5C								
GIODSETB	0x60								
GIODCLRB	0x64								
GIOPDRB	0x68								
GIOPULDISB	0x6C								
GIOPSLB	0x70								
GIODIRC	0x74								
GIODINC	0x78								
GIODOUTC	0x7C								
GIODSETC	0x80								
GIODCLRC	0x84								
GIOPDRC	0x88								
GIOPULDISC	0x8C								
GIOPSLC	0x90								
GIODIRD	0x94								
GIODIND	0x98								
GIODOUTD	0x9C								
GIODSETD	0xA0								
GIODCLRD	0xA4								
GIOPDRD	0xA8								

Table 19-26. Example GIO Register Set Showing Reserved Bits (Continued)

GIOPULDISD	0xAC							
GIOPSLD	0xB0							
GIODIRE	0xB4							
GIODINE	0xB8							
GIODOUTE	0xBC							
GIODSETE	0xC0							
GIODCLRE	0xC4							
GIOPDRE	0xC8							
GIOPULDISE	0xCC							
GIOPSLE	0xD0							
GIODIRF	0xD4							
GIODINF	0xD8							
GIODOUTF	0xDC							
GIODSETF	0xE0							
GIODCLRF	0xE4							
GIOPDRF	0xE8							
GIOPULDISF	0xEC							
GIOPSLF	0xF0							
GIODIRG	0xF4							
GIODING	0xF8							
GIODOUTG	0xFC							
GIODSETG	0x100							
GIODCLRG	0x104							
GIOPDRG	0x108							
GIOPULDISG	0x10C							
GIOPSLG	0x110							
GIODIRH	0x114							
GIODINH	0x118							
GIODOUTH	0x11C							
GIODSETH	0x120							
GIODCLRH	0x124							
GIOPDRH	0x128							
GIOPULDISH	0x12C							
GIOPSLH	0x130							

19.7.1 Example: Setting Interrupts and Configuring Pins for Output

The following code demonstrates how to set the interrupts. In the example, two of the interrupts are enabled, and the third pin of port A is configured for output. R2 keeps the same value, and the other registers act as temporary storage for addresses and values.

```
GIO_LOC     .word 0xFFF7BC00       ;device specific address for the GIO registers.
GIOENASET   .equ     0x0C            ;setting up equate statements
GIOPOL      .equ    0x08
GIOFLG      .equ    0x1C
GIOPRYSET   .equ    0x14
```

```
GIOPRYCLR  .equ    0x18
GIOOFFA    .equ    0x20
GIOEMUA    .equ    0x28
GIOOFFB    .equ    0x24
GIOEMUB    .equ    0x2C
GIODIRA    .equ    0x30
GIODINA    .equ    0x34
GIODOUTA   .equ    0x38
GIODSETA   .equ    0x3C
GIODCLRA   .equ    0x40
GIOPDRA    .equ    0x44
GIOPULDISA .equ    0x48


   LDR R2, GIO_LOC              ;loads GIO base address into register 2.
                                ; **SET THE POLARITY OF THE INTERRUPTS.**
   MOV R3, #GIOPOL              ; loads GIOPOL offset address into register 3.
   MOV R4, #0x02                ; loads a value of 0x02 into register 4.
                                ; Sets bit 1.
   STR R4, [R2, R3]             ; sets interrupt 0 to trigger on falling edge
                                ; and interrupt 1 to trigger on rising edge.
                                ; **SET THE PRIORITY ON BOTH INTERRUPTS AS LOW**
   MOV R3, #GIOPRYCLR           ; loads the GIOPRY address into register 3.
   MOV R4, #0x03                ; loads a value of 0 into register 4.
   STR R4, [R2, R3]             ; priority on interrupts 0 and 1 is set LOW.
                                ; ** SET PORT A FOR OUTPUT (EXCEPT FOR INTERRUPTS
                                ; WHICH MUST BE CONFIGURED AS INPUTS)**
   MOV R3, #GIODIRA             ; loads GIODIRA offset address into register 3.
   MOV R4, #0xFC                ; loads a binary value of 11111100
   STR R4, [R2, R3]             ; sets pins 0 and 1 as input to insure proper
                                ; interrupt behavior pin 2 configured as
                                ; output
   MOV R3, #GIOFLG              ; loads GIOFLG offset address into register 3.
   MOV R4, #0xFF                ; sets 32 bits.
   STR R4, [R2, R3]             ; clears all bits of the flag register.
 ; **ENABLE THE FIRST TWO INTERRUPTS 1:0.**
  MOV R3, #GIOENA ; loads GIOENA offset address into register 3.
  MOV R4, #0x03 ; sets first 2 bits 1:0
  STR R4, [R2, R3] ; enables the first two interrupts
```

19.7.2 Example: Toggling Output Buffers

The following code demonstrates how to toggle the output buffer port B using the GIODSET and GIODCLR buffers.

```
GIO_LOC    .word   0xFFF7BC00   ;device specific address for the GIO registers.
GIOENA     .equ    0x04       ;setting up equate statements
...
```

```
GIODIRB  .equ    0x58
GIODINB   .equ    0x5C
GIODOUTB  .equ    0x60
GIODSETB  .equ    0x64
GIODCLRB  .equ    0x68
   LDR R2, GIO_LOC              ; loads absolute address of the GIO memory
                                ; location (device specific).
                                ; **SET ALL BITS IN GIODOUTB AS 1.**
   MOV R3, #GIODSETB            ; loads offset address of GIODSETB
   MOV R4, #0xFF                ; loads a binary value of 11111111
   STR R4, [R2, R3]             ; sets all bits of GIODOUTB.
                                ; **CONFIGURE PORT 1 AS OUTPUT.**
   MOV R3, #GIODIRB             ; loads offset address of GIODIRB.
   MOV R4, #0xFF                ; loads a binary value of 11111111.
   STR R4, [R2, R3]             ; configures port 1 as output
                                ; **TOGGLE BITS 0, 2, 4, 6.**
   MOV R3, #GIODSETB            ; loads offset address of GIODSETB
   MOV R4, #GIODCLRB            ; loads GIODCLRB offset address
   MOV R5, #0x55                ; loads a binary value of 01010101
TOGGLE
   STR R5, [R2, R4]             ; clears GIODOUTB bits 0, 2, 4, 6
   STR R5, [R2, R3]             ; sets GIODOUTB bits 0, 2, 4, 6
     B TOGGLE        ; loops back to TOGGLE
```

19.7.3 Example: Clearing Interrupt Flags and Setting Interrupts

The following code demonstrates how interrupt flags should be cleared before interrupts are enabled. In this example, all three interrupt-capable pins are set as interrupts.

```
GIO_LOC     .word   0xFFF7BC00   ;device specific address for GIO registers
GIOENASET   .equ    0x0C       ;setting up equate statements
...
GIOFLG      .equ    0x1C



   LDR R2, GIO_LOC              ;loads absolute address of the GIO memory
   MOV R3, #GIOFLG              ;loads GIOFLG offset address into R3.
   MOV R4, #0x07                ;loads a value of 00000111 into R4.
   STR R4, [R2, R3]             ;clears the interrupt-capable bits of the
                                ;GIOFLG control register.

   MOV R3, #GIOENA              ;loads GIOENA offset address into R3.
   MOV R4, #0x07                ;loads 00000111 into R4.
   STR R4, [R2, R3]             ;enables pins 2:0 of port A as interrupts.
```

19.7.4 Example: Reading Port B Input Register

The following code demonstrates how to read the input register of port A when interrupts are enabled. Pin 0 is set as an interrupt, and pins 2 and 1 are configured as inputs.

```
GIO_LOC.word 0xFFF7BC00          ;device specific address for GIO registers
GIOENA .equ    0x04              ;setting up equate statements
...
GIODIRB .equ 0x58
GIODINB .equ 0x5C
GIODOUTB.equ 0x60
GIODSETB.equ 0x64
GIODCLRB .equ 0x68
   LDR R2, GIO_LOC               ;loads absolute address of the GIO memory
                                 ;**MASK OUTPUTS AND INTERRUPTS SO INPUT IS UNAMBIGUOUS.**
   MOV R3, #GIODINB              ;loads the offset address of GIODINB.
   LDR R4, [R2, R3]              ;loads the value in GIODINB register into R4.
   MOV R3, #0xFE                 ;loads 11111110 into R3. This value is used
                                 ;to mask the input so that only the input
                                 ;values are read. The 1's appear in the places
                                 ;where the input register is reading input
                                 ;voltages.
   AND R4, R4, R3                ;loads masked input value into R4.
```

Revision History

This revision history highlights the technical changes made to the device or the Technical Reference Manual (TRM).

Date	Additions, Deletions, And Modifications	Revision
March 2010		0
October 2010	Remove the slew rate control from every module. Add RTI clock ratio notes to Section 4.2.21. Updated section 16.2.3.1, *DWD*, page 860, the DWD is disabled by default. Added section 16.3.25, *Digital Watchdog Control Register (RTIDWDCTRL)*, page 892. Updated section 16.3.27, *Watchdog Status Register (RTIWDSTATUS)*, page 894. Updated Table 4-1, swapped the offset of MSTCGSTAT and MSTFAIL. Updated the HET base address to 0xFFF7B800 in section 12.6, *HET Control Registers*, page 624	A

Date	Additions, Deletions, And Modifications	Revision
November 2010	Errata Update BTS_DCAN_18, add a note in Page 382; BTS_DCAN_19, add a note in Page 384 and bit 29 description in Table 10-19 BTS_DCAN_21, Section 10.17.18 BTS_DCAN_23, Table 10-20 BTS_ESM_12, No change BTS_ESRAMW_29, No change, section 5.7, *Control Registers*, page 193 BTS_ESRAMW_35, shade bit 15, Figure 5-14 BTS_FWM_90, No change to section 7.4.1.3, *Read Margin Modes*, page 228 BTS_FWM_154, No change to section 7.5.16, *Flash Pump Access Control Register 1 (FPAC1 - 0xFFF87048)*, page 264 BTS_GIO_9, No change to section 19.6.11, *GIO Data Direction Registers [A-H][7:0] (GIODIR[A-H][7:0])*, page 1084 BTS_HET_16, add a note in Page 603 BTS_LIN_47, add a note in Page 947 BTS_LIN_52, No change, it is included in section 18.8.8.2, *Physical Bus Errors*, page 954 BTS_LIN_53, add a note in Page 953 BTS_LIN_54, No change, it is included in the note in Page 965 BTS_M3VIM_5, No change, it is included in section 8.6.1, *M3VIM Channel Offset Registers*, page 290 BTS_M3VIM_8, No change, it is included in section 8.5.5, *Reset*, page 283 BTS_M3VIM_11, No change, it is included in section 8.6.6, *Interrupt Mask Set Registers (REQMASKSETx)*, page 302 BTS_MIBSPI_89, No change, it is included in Table 11-50 BTS_MIBSPI_113, No change, it is included in the note in Page 570 BTS_MIBSPI_114, No change, it is included in the Table 11-52 BTS_MIBSPI_124/125, fixed in the datasheet BTS_SSW_25, make change in section 6.3.1.4, *Frequency Modulation*, page 212 BTS_SSW_26, not applicable to user manual, reserved bits. BTS_STC_16, user manual is correct, no change BTS_SYS_51/59, user manual is correct, no change BTS_SYS_74, add a note in Table 4-54	A

Date	Additions, Deletions, And Modifications	Revision
January 2012	Remove “The number of patterns per interval is fixed to 32.” from Page 718. Change STC base address from 0xFFFFF800 to 0xFFFFE400, Page 728 Removed RTP information from Figure 1-2, Table 4-1, Figure 5-1,Section 5.2, Section 5.3, Table 5-15. Added ECPCLK2 description in section 4.4.2, *ECP Control Register (ECPCNTL1)*, page 176 and section 4.4.3, *ECP Control Register (ECPCNTL2)*, page 177 for some configurations of TMS470M devices. Modified section 4.2.1, *SYS Pin Control Register 1 (SYSPC1)*, page 66 ~ section 4.2.9, *SYS Pin Control Register 9 (SYSPC9)*, page 76 for some configurations of TMS470M devices. Modified bit 8 and bit 9 description in register Table 7-6 (section 7.5.3, *Flash Error Detection and Correction Control Register 1 (FEDACCTRL1 - 0xFFF87008)*, page 247). Removed ‘PBIST’ in Table 4-29 since only MBIST is available in this device. Added a note in Page 738. Replaced 20Mhz baud rate to 11Mhz baud rate in section 11.1, *Overview*, page 446. Replaced disticted device name to TMS470M in Table 15-13. Renamed MSINENA to MSIENA throughout the document to make it consistent. Add section 1.10.3, *Memory BIST Algorithms*, page 40. Switched the position of R1 and R2 in Figure 15-12 and Figure 15-15. Add a note in Page 1061 to address the ‘nTRST override input buffer disable’ issue. Modified the note in Page 1061 to expand it to other peripheral pins when used as normal GIO pin. Remove ‘DMA’ from chapters 1-9. BTS_DCC_23, no change since it is defined as test module. BTS_ESRAMW_32, add description at the end of section 5.6, *Hardware RAM Initialization*, page 192. BTS_FWM_149, Correct the description of FCOR_ERR_CNT in Table 7-8. BTS_SYS_72, replace power-on reset with reset in Figure 4-32. BTS_SYS_87, Modify the description in Table 4-48. BTS_SYS_92, Modify the max value from 3Fh to FFh in Table 4-41. BTS_SYS_93, Remove bit 2 in section 4.2.53, *System Exception Status Register (SYSESR)*, page 133. BTS_SYS_94, Modify description of sleep mode in Table 4-34. BTS_SYS_96, no change since PRTY_PBM bit is not available on this device. BTS_SYS_109, no change, this document is correct. BTS_TMS470M_2, no change, this document is correct. BTS_TMS470M_8, no change, mentioned in datasheet. BTS_TMS470M_9, no change, mentioned in datasheet. Removed **‘because of the slow output buffers used’** from Table 4-84 and Table 4-85. Add notes on Page 176, Page 177, Page 66, Page 67, Page 68, Page 69, Page 71, Page 72, Page 73, Page 75, Page 76, to clarity that ECPCLK2 is only available on some configurations of TMS470M series.	B
March 2013	Removed sections related to Hibernate mode and External Wake Up 2.2.4 - External Wake Up Signals 2.2.5 - Wakeup Registers 4.4 - Wakeup Control Registers	C

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2013, Texas Instruments Incorporated