

*Preliminary*

# **RM48 16/32-Bit RISC Flash Microcontroller**

## **Technical Reference Manual**



Literature Number: SPNU499  
September 2011

*Preliminary*

<b>Preface</b> .....	<b>83</b>
<b>1 Introduction</b> .....	<b>85</b>
1.1 Designed for Safety Applications .....	86
1.2 Family Description .....	86
1.3 Endianism Considerations .....	89
1.3.1 RM48x: Little Endian (LE) .....	89
<b>2 Architecture</b> .....	<b>91</b>
2.1 Introduction .....	92
2.1.1 Architecture Block Diagram .....	92
2.1.2 Definitions of Terms .....	93
2.1.3 Bus Master / Slave Access Privileges .....	95
2.1.4 Device Endianness .....	95
2.2 Clocks .....	96
2.2.1 Clock Sources .....	96
2.2.2 Clock Domains .....	97
2.2.3 Low Power Modes .....	98
2.2.4 Clock Test Mode .....	100
2.2.5 Safety Considerations for Clocks .....	101
2.3 Memory Organization .....	103
2.3.1 Memory Map Overview .....	103
2.3.2 Memory Map Table .....	104
2.3.3 Flash on Microcontrollers .....	108
2.3.4 On-Chip SRAM .....	110
2.4 Exceptions .....	115
2.4.1 Resets .....	115
2.4.2 Aborts .....	115
2.4.3 System Software Interrupts .....	117
2.5 System and Peripheral Control Registers .....	118
2.5.1 Primary System Control Registers (SYS) .....	118
2.5.2 Secondary System Control Registers (SYS2) .....	182
2.5.3 Peripheral Central Resource (PCR) Control Registers .....	194
<b>3 Power Management Module (PMM)</b> .....	<b>219</b>
3.1 Overview .....	220
3.1.1 Main Features of the Power Management Module (PMM) .....	220
3.1.2 Block Diagram .....	220
3.2 Power Domains .....	222
3.3 PMM Operation .....	223
3.3.1 Power Switch .....	223
3.3.2 Power Domain State .....	223
3.3.3 Default Power Domain State .....	223
3.3.4 Disabling a Power Domain Permanently .....	223
3.3.5 Changing Power Domain State .....	223
3.3.6 Reset Management .....	224
3.3.7 Diagnostic Power State Controller (PSCON) .....	224
3.3.8 PSCON Compare Block .....	224

3.4	PMM Registers .....	226
3.4.1	Logic Power Domain Control Register (LOGICPDPWRCTRL0) .....	227
3.4.2	Memory Power Domain Control Register 0 (MEMPDWRCTRL0) .....	229
3.4.3	Power Domain Clock Disable Register (PDCLKDISREG) .....	230
3.4.4	Power Domain Clock Disable Set Register (PDCLKDISSETREG) .....	231
3.4.5	Power Domain Clock Disable Clear Register (PDCLKDISCLRREG) .....	232
3.4.6	Logic Power Domain PD2 Power Status Register (LOGICPDPWRSTAT0) .....	233
3.4.7	Logic Power Domain PD3 Power Status Register (LOGICPDPWRSTAT1) .....	234
3.4.8	Logic Power Domain PD4 Power Status Register (LOGICPDPWRSTAT2) .....	235
3.4.9	Logic Power Domain PD5 Power Status Register (LOGICPDPWRSTAT3) .....	236
3.4.10	Memory Power Domain RAM_PD1 Power Status Register (MEMPDWRSTAT0) .....	237
3.4.11	Memory Power Domain RAM_PD2 Power Status Register (MEMPDWRSTAT1) .....	238
3.4.12	Memory Power Domain RAM_PD3 Power Status Register (MEMPDWRSTAT2) .....	239
3.4.13	Global Control Register 1 (GLOBALCTRL1) .....	240
3.4.14	Global Status Register (GLOBALSTAT) .....	241
3.4.15	PSCON Diagnostic Compare Key Register (PRCKEYREG) .....	242
3.4.16	LogicPD PSCON Diagnostic Compare Status Register 1 (LPDDCSTAT1) .....	243
3.4.17	LogicPD PSCON Diagnostic Compare Status Register 2 (LPDDCSTAT2) .....	244
3.4.18	Memory PD PSCON Diagnostic Compare Status Register 1 (MPDDCSTAT1) .....	245
3.4.19	Memory PD PSCON Diagnostic Compare Status Register 2 (MPDDCSTAT2) .....	246
3.4.20	Isolation Diagnostic Status Register (ISODIAGSTAT) .....	247
<b>4</b>	<b>I/O Multiplexing and Control Module (IOMM) .....</b>	<b>249</b>
4.1	Overview .....	250
4.2	Main Features of I/O Multiplexing Module (IOMM) .....	250
4.3	Control of Multiplexed Functions .....	251
4.3.1	Control of Multiplexed Outputs .....	251
4.3.2	Control of Multiplexed Inputs .....	252
4.3.3	Control of Special Multiplexed Options .....	253
4.4	Safety Features .....	254
4.4.1	Locking Mechanism for Memory-Mapped Registers .....	254
4.4.2	Master ID Check .....	254
4.4.3	Error Conditions .....	254
4.5	IOMM Registers .....	255
4.5.1	REVISION_REG: Revision Register .....	256
4.5.2	BOOT_REG: Boot Mode Register .....	257
4.5.3	KICK_REG0: Kicker Register 0 .....	258
4.5.4	KICK_REG1: Kicker Register 1 .....	259
4.5.5	ERR_RAW_STATUS_REG: Error Raw Status / Set Register .....	260
4.5.6	ERR_ENABLED_STATUS_REG: Error Enabled Status / Clear Register .....	261
4.5.7	ERR_ENABLE_REG: Error Signaling Enable Register .....	262
4.5.8	ERR_ENABLE_CLR_REG: Error Signaling Enable Clear Register .....	263
4.5.9	FAULT_ADDRESS_REG: Fault Address Register .....	264
4.5.10	FAULT_STATUS_REG: Fault Status Register .....	265
4.5.11	FAULT_CLEAR_REG: Fault Clear Register .....	266
4.5.12	PINMMRnn: Pin Multiplexing Control Registers .....	267
4.6	Signal Multiplexing and Control on Microcontrollers .....	268
<b>5</b>	<b>F021 Flash Module .....</b>	<b>273</b>
5.1	Overview .....	274
5.1.1	Features .....	274
5.1.2	Definition of Terms .....	274
5.1.3	F021 Flash Tools .....	275
5.2	Default Flash Configuration .....	276
5.3	Memory Map .....	276



5.4	Control Registers .....	277
5.4.1	Flash Option Control Register (FRDCNTL) .....	278
5.4.2	Flash Error Detection and Correction Control Register 1 (FEDACCTRL1) .....	279
5.4.3	Flash Error Correction and Correction Control Register 2 (FEDACCTRL2) .....	281
5.4.4	Flash Un-correctable Error Address (FUNC_ERR_ADD) .....	282
5.4.5	Flash Bank Protection Register (FBPROT) .....	283
5.4.6	Flash Bank Sector Enable Register (FBSE) .....	284
5.4.7	Flash Bank Access Control Register (FBAC) .....	285
5.4.8	Flash Bank Fallback Power Register (FBFALLBACK) .....	286
5.4.9	Flash Pump Access Control Register 1 (FPAC1) .....	287
5.4.10	Flash Pump Access Control Register 2 (FPAC2) .....	288
5.4.11	Flash Module Access Control Register (FMAC) .....	289
5.4.12	Flash Module Status Register (FMSTAT) .....	290
5.4.13	FSM Register Write Enable (FSM_WR_ENA) .....	292
5.4.14	EEPROM Emulation Configuration Register (EEPROM_CONFIG) .....	293
<b>6</b>	<b>Tightly-Coupled RAM (TCRAM) Module .....</b>	<b>295</b>
6.1	Overview .....	296
6.1.1	B0TCM and B1TCM Connection Diagram .....	296
6.1.2	Main Features .....	296
6.2	RAM Memory Map .....	298
6.3	Safety Features .....	299
6.3.1	Support for Cortex-R4F CPU's Single-Error-Correction Double-Error-Detection (SECDED) .....	299
6.3.2	Support for Cortex-R4F CPU's Address and Control Bus Parity Checking .....	299
6.3.3	Redundant Address Decode .....	300
6.4	TCRAM Auto-Initialization .....	301
6.5	Trace Module Support .....	301
6.6	Emulation / Debug Mode Behavior .....	301
6.7	Control and Status Registers .....	302
6.7.1	TCRAM Module Control Register (RAMCTRL) .....	303
6.7.2	TCRAM Module Single-Bit Error Correction Threshold Register (RAMTHRESHOLD) .....	305
6.7.3	TCRAM Module Single-Bit Error Occurrences Counter Register (RAMOCCUR) .....	306
6.7.4	TCRAM Module Interrupt Control Register (RAMINTCTRL) .....	307
6.7.5	TCRAM Module Error Status Register (RAMERRSTATUS) .....	308
6.7.6	TCRAM Module Single-Bit Error Address Register (RAMSERRADDR) .....	309
6.7.7	TCRAM Module Uncorrectable Error Address Register (RAMUERRADDR) .....	310
6.7.8	TCRAM Module Test Mode Control Register (RAMTEST) .....	311
6.7.9	TCRAM Module Test Mode Vector Register (RAMADDRDECVECT) .....	312
6.7.10	TCRAM Module Parity Error Address Register (RAMPERRADDR) .....	313
<b>7</b>	<b>Programmable Built-In Self-Test (PBIST) Module .....</b>	<b>315</b>
7.1	Overview .....	316
7.1.1	Features of PBIST .....	316
7.1.2	PBIST vs. Application Software-Based Testing .....	316
7.1.3	PBIST Block Diagram .....	316
7.2	RAM Grouping and Algorithm .....	318
7.3	PBIST Flow .....	321
7.3.1	PBIST Sequence .....	322
7.4	Memory Test Algorithms on the On-chip ROM .....	324
7.5	PBIST Control Registers .....	326
7.5.1	RAM Configuration Register (RAMT) .....	327
7.5.2	Datalogger Register (DLR) .....	328
7.5.3	PBIST Activate/ROM Clock Enable Register (PACT) .....	329
7.5.4	PBIST ID Register .....	330
7.5.5	Override Register (OVER) .....	331

7.5.6	Fail Status Fail Registers (FSRF0 and FSRF1)	332
7.5.7	Fail Status Count Registers (FSRC0 and FSRC1)	333
7.5.8	Fail Status Address Registers (FSRA0 and FSRA1)	334
7.5.9	Fail Status Data Registers (FSRDL0 and FSRDL1)	335
7.5.10	ROM Mask Register (ROM)	336
7.5.11	ROM Algorithm Mask Register (ALGO)	337
7.5.12	RAM Info Mask Lower Register (RINFOL)	338
7.5.13	RAM Info Mask Upper Register (RINFOU)	339
7.6	PBIST Configuration Example	340
7.6.1	Example 1 : Configuration of PBIST Controller to Run Self-Test on DCAN1 RAM	340
7.6.2	Example 2 : Configuration of PBIST Controller to Run Self-Test on ALL RAM Groups	340
<b>8</b>	<b>CPU Self-Test Controller (STC) Module</b>	<b>343</b>
8.1	General Description	344
8.1.1	CPU Self-Test Controller Features	344
8.1.2	STC Block Diagram	344
8.2	Application Self-test Flow	346
8.2.1	STC Module Configuration	346
8.2.2	Context Saving	346
8.2.3	Entering CPU Idle Mode	346
8.2.4	Self-Test Completion and Error Generation	347
8.3	STC Test Coverage and Duration	348
8.4	STC Control Registers	349
8.4.1	STC Global Control Register0 (STCGCR0)	350
8.4.2	STC Global Control Register1 (STCGCR1)	351
8.4.3	Self-Test Run Timeout Counter Preload Register (STCTPR)	352
8.4.4	STC Current ROM Address Register (STC_CADDR)	353
8.4.5	STC Current Interval Count Register (STCCICR)	354
8.4.6	Self-Test Global Status Register (STCGSTAT)	355
8.4.7	Self-Test Fail Status Register (STCFSTAT)	356
8.4.8	CPU1 Current MISR Register (CPU1_CURMISR[3:0])	357
8.4.9	CPU2_CURMISR[3:0] (CPU2 Current MISR Register)	359
8.5	STC Configuration Example	361
<b>9</b>	<b>CPU Compare Module for Cortex™-R4F (CCM-R4F)</b>	<b>363</b>
9.1	Main Features	364
9.2	Block Diagram	364
9.3	Module Operation	365
9.3.1	1001D Lock Step Mode	365
9.3.2	Self-Test Mode	365
9.3.3	Error Forcing Mode	367
9.3.4	Self-Test Error Forcing Mode	367
9.3.5	Operation During CPU Debug Mode	367
9.4	Control Registers	368
9.4.1	CCM-R4F Status Register (CCMSR)	369
9.4.2	CCM-R4F Key Register (CCMKEYR)	370
<b>10</b>	<b>Oscillator and PLL</b>	<b>371</b>
10.1	Introduction	372
10.1.1	Features	372
10.2	Quick Start	373
10.3	Oscillator	374
10.3.1	Oscillator Implementation	375
10.3.2	Oscillator Enable	375
10.3.3	Oscillator Disable	375

10.4	Low Power Oscillator and Clock Detect (LPOCLKDET)	376
10.4.1	Clock Detect	376
10.4.2	Behavior on Oscillator Failure	376
10.4.3	Recovery from Oscillator Failure	377
10.4.4	LPOCLKDET Enable	377
10.4.5	LPOCLKDET Disable	377
10.4.6	Trimming the HF LPO Oscillator	378
10.5	PLL	379
10.5.1	Modulation	380
10.5.2	PLL Output Control	381
10.5.3	Behavior on PLL Fail	383
10.5.4	Recovery From a PLL Failure	384
10.5.5	PLL Modulation Depth Measurement	384
10.5.6	PLL Frequency Measurement Circuit	385
10.5.7	PLL2	385
10.6	Control Registers	386
10.6.1	PLL Control 1 Register (PLLCTL1)	387
10.6.2	PLL Control 2 Register (PLLCTL2)	389
10.6.3	PLL Control 3 Register (PLLCTL3)	391
10.6.4	PLL1 Clock Slip Control Register (CLKSLIP)	393
10.6.5	General Purpose Register1 (GPREG1)	394
10.6.6	PLL Modulation Depth Measurement Control Register (SSWPLL1)	395
10.6.7	SSW PLL BIST Control Register 2 (SSWPLL2)	397
10.6.8	SSW PLL BIST Control Register 3 (SSWPLL3)	398
10.6.9	LPO/Clock Monitor Control Register (LPOMONCTL)	399
10.6.10	Clock Test Register (CLKTEST)	402
10.7	Phase-Locked Loop Theory of Operation	403
10.7.1	Phase-Frequency Detector	403
10.7.2	Charge Pump and Loop Filter	404
10.7.3	Voltage-Controlled Oscillator	404
10.7.4	Frequency Modulation	404
10.8	Programming Example	406
<b>11</b>	<b>Dual-Clock Comparator (DCC) Module</b>	<b>409</b>
11.1	Introduction	410
11.1.1	Main Features	410
11.1.2	Block Diagram	410
11.2	Module Operation	411
11.2.1	Error Conditions	411
11.2.2	Single-Shot Operation Mode	413
11.3	Clock Source Selection for Counter0 and Counter1	414
11.3.1	Clock Sources for DCC1	414
11.3.2	Clock Sources for DCC2	414
11.4	DCC Control Registers	415
11.4.1	DCC Global Control Register (DCCGCTRL)	416
11.4.2	DCC Revision Id Register (DCCREV)	417
11.4.3	DCC Counter0 Seed Register (DCCCNT0SEED)	418
11.4.4	DCC Valid0 Seed Register (DCCVALID0SEED)	419
11.4.5	DCC Counter1 Seed Register (DCCCNT1SEED)	420
11.4.6	DCC Status Register (DCCSTAT)	421
11.4.7	DCC Counter0 Value Register (DCCCNT0)	422
11.4.8	DCC Valid0 Value Register (DCCVALID0)	423
11.4.9	DCC Counter1 Value Register (DCCCNT1)	424
11.4.10	DCC Counter1 Clock Source Selection Register (DCCCNT1CLKSRC)	425

11.4.11	DCC Counter0 Clock Source Selection Register (DCCNT0CLKSRC)	426
<b>12</b>	<b>Error Signaling Module (ESM)</b>	<b>427</b>
12.1	Overview	428
12.1.1	Feature List	428
12.1.2	Block Diagram	428
12.2	Module Operation	430
12.2.1	Reset Behavior	430
12.2.2	ERROR Pin Timing	430
12.2.3	Forcing an Error Condition	432
12.3	Recommended Programming Procedure	433
12.4	Control Registers	434
12.4.1	ESM Enable ERROR Pin Action/Response Register 1 (ESMEEPAPR1)	435
12.4.2	ESM Disable ERROR Pin Action/Response Register 1 (ESMDEPAPR1)	436
12.4.3	ESM Interrupt Enable Set/Status Register 1 (ESMIESR1)	437
12.4.4	ESM Interrupt Enable Clear/Status Register 1 (ESMIECR1)	438
12.4.5	ESM Interrupt Level Set/Status Register 1 (ESMILSR1)	439
12.4.6	ESM Interrupt Level Clear/Status Register 1 (ESMILCR1)	440
12.4.7	ESM Status Register 1 (ESMSR1)	441
12.4.8	ESM Status Register 2 (ESMSR2)	442
12.4.9	ESM Status Register 3 (ESMSR3)	443
12.4.10	ESM ERROR Pin Status Register (ESMEPSR)	444
12.4.11	ESM Interrupt Offset High Register (ESMIOFFHR)	445
12.4.12	ESM Interrupt Offset Low Register (ESMIOFFLR)	446
12.4.13	ESM Low-Time Counter Register (ESMLTCR)	447
12.4.14	ESM Low-Time Counter Preload Register (ESMLTCPR)	448
12.4.15	ESM Error Key Register (ESMEKR)	449
12.4.16	ESM Status Shadow Register 2 (ESMSSR2)	450
12.4.17	ESM Influence ERROR Pin Set/Status Register 4 (ESMIEPSR4)	451
12.4.18	ESM Influence ERROR Pin Clear/Status Register 4 (ESMIEPCR4)	452
12.4.19	ESM Interrupt Enable Set/Status Register 4 (ESMIESR4)	453
12.4.20	ESM Interrupt Enable Clear/Status Register 4 (ESMIECR4)	454
12.4.21	ESM Interrupt Level Set/Status Register 4 (ESMILSR4)	455
12.4.22	ESM Interrupt Level Clear/Status Register 4 (ESMILCR4)	456
12.4.23	ESM Status Register 4 (ESMSR4)	457
<b>13</b>	<b>Real-Time Interrupt (RTI) Module</b>	<b>459</b>
13.1	Overview	460
13.1.1	Features	460
13.1.2	Industry Standard Compliance Statement	460
13.2	Module Operation	461
13.2.1	Counter Operation	461
13.2.2	Interrupt/DMA Requests	463
13.2.3	RTI Clocking	464
13.2.4	Synchronizing Timer Events to Network Time (NTU)	464
13.2.5	Digital Watchdog (DWD)	467
13.2.6	Low Power Modes	469
13.2.7	Halting Debug Mode Behaviour	470
13.3	Control Registers	471
13.3.1	RTI Global Control Register (RTIGCTRL)	473
13.3.2	RTI Timebase Control Register (RTITBCTRL)	474
13.3.3	RTI Capture Control Register (RTICAPCTRL)	475
13.3.4	RTI Compare Control Register (RTICOMPCTRL)	476
13.3.5	RTI Free Running Counter 0 Register (RTIFRC0)	477
13.3.6	RTI Up Counter 0 Register (RTIUC0)	478

13.3.7	RTI Compare Up Counter 0 Register (RTICPUC0)	479
13.3.8	RTI Capture Free Running Counter 0 Register (RTICAFRC0)	480
13.3.9	RTI Capture Up Counter 0 Register (RTICAUC0)	481
13.3.10	RTI Free Running Counter 1 Register (RTIFRC1)	482
13.3.11	RTI Up Counter 1 Register (RTIUC1)	483
13.3.12	RTI Compare Up Counter 1 Register (RTICPUC1)	484
13.3.13	RTI Capture Free Running Counter 1 Register (RTICAFRC1)	485
13.3.14	RTI Capture Up Counter 1 Register (RTICAUC1)	486
13.3.15	RTI Compare 0 Register (RTICOMP0)	487
13.3.16	RTI Update Compare 0 Register (RTIUDCP0)	488
13.3.17	RTI Compare 1 Register (RTICOMP1)	489
13.3.18	RTI Update Compare 1 Register (RTIUDCP1)	490
13.3.19	RTI Compare 2 Register (RTICOMP2)	491
13.3.20	RTI Update Compare 2 Register (RTIUDCP2)	492
13.3.21	RTI Compare 3 Register (RTICOMP3)	493
13.3.22	RTI Update Compare 3 Register (RTIUDCP3)	494
13.3.23	RTI Timebase Low Compare Register (RTITBLCOMP)	495
13.3.24	RTI Timebase High Compare Register (RTITBHCOMP)	496
13.3.25	RTI Set Interrupt Enable Register (RTISETINTENA)	497
13.3.26	RTI Clear Interrupt Enable Register (RTICLEARINTENA)	499
13.3.27	RTI Interrupt Flag Register (RTIINTFLAG)	501
13.3.28	Digital Watchdog Control Register (RTIDWDCTRL)	503
13.3.29	Digital Watchdog Preload Register (RTIDWDPRLD)	504
13.3.30	Watchdog Status Register (RTIWDSTATUS)	505
13.3.31	RTI Watchdog Key Register (RTIWDKEY)	506
13.3.32	RTI Digital Watchdog Down Counter (RTIDWDCNTR)	507
13.3.33	Digital Windowed Watchdog Reaction Control (RTIWWDRXNCTRL)	508
13.3.34	Digital Windowed Watchdog Window Size Control (RTIWWDSIZECTRL)	509
<b>14</b>	<b>Cyclic Redundancy Check Controller (CRC) Module</b>	<b>511</b>
14.1	Overview	512
14.1.1	Features	512
14.2	Module Operation	515
14.2.1	General Operation	515
14.2.2	CRC Modes of Operation	515
14.2.3	PSA Signature Register	516
14.2.4	PSA Sector Signature Register	517
14.2.5	CRC Value Register	517
14.2.6	Raw Data Register	518
14.2.7	Example DMA Controller Setup	518
14.2.8	Pattern Count Register	520
14.2.9	Sector Count Register/Current Sector Register	520
14.2.10	Interrupt	521
14.2.11	CPU Data Trace	524
14.2.12	Power Down Mode	525
14.2.13	Emulation	525
14.2.14	Peripheral Bus Interface	525
14.3	Example	526
14.3.1	Example: Auto Mode Using Time Based Event Triggering	526
14.3.2	Example: Auto Mode Without Using Time Based Triggering	526
14.3.3	Example: Semi-CPU Mode	527
14.3.4	Example: Full-CPU Mode	528
14.4	CRC Control Registers	529
14.4.1	CRC Global Control Register 0 (CRC_CTRL0)	531

14.4.2	CRC Global Control Register (CRC_CTRL1) .....	532
14.4.3	CRC Global Control Register 2 (CRC_CTRL2) .....	533
14.4.4	CRC Interrupt Enable Set Register (CRC_INTS) .....	535
14.4.5	CRC Interrupt Enable Reset Register (CRC_INTR) .....	540
14.4.6	CRC Interrupt Status Register (CRC_STATUS) .....	545
14.4.7	CRC Interrupt Offset (CRC_INT_OFFSET_REG) .....	548
14.4.8	CRC Busy Register (CRC_BUSY) .....	549
14.4.9	CRC Pattern Counter Preload Register 1 (CRC_PCOUNT_REG1) .....	550
14.4.10	CRC Sector Counter Preload Register 1 (CRC_SCOUNT_REG1) .....	551
14.4.11	CRC Current Sector Register 1 (CRC_CURSEC_REG1) .....	552
14.4.12	CRC Channel 1 Watchdog Timeout Preload Register A (CRC_WDTPLD1) .....	553
14.4.13	CRC Channel 1 Block Complete Timeout Preload Register B (CRC_BCTOPLD1) .....	554
14.4.14	Channel 1 PSA Signature Low Register (PSA_SIGREGL1) .....	555
14.4.15	Channel 1 PSA Signature High Register (PSA_SIGREGH1) .....	556
14.4.16	Channel 1 CRC Value Low Register (CRC_REGL1) .....	557
14.4.17	Channel 1 CRC Value High Register (CRC_REGH1) .....	558
14.4.18	Channel 1 PSA Sector Signature Low Register (PSA_SECSIGREGL1) .....	559
14.4.19	Channel 1 PSA Sector Signature High Register (PSA_SECSIGREGH1) .....	560
14.4.20	Channel 1 Raw Data Low Register (RAW_DATAREGL1) .....	561
14.4.21	Channel 1 Raw Data High Register (RAW_DATAREGH1) .....	562
14.4.22	CRC Pattern Counter Preload Register 2 (CRC_PCOUNT_REG2) .....	563
14.4.23	CRC Sector Counter Preload Register 2 (CRC_SCOUNT_REG2) .....	564
14.4.24	CRC Current Sector Register 2 (CRC_CURSEC_REG2) .....	565
14.4.25	CRC Channel 2 Watchdog Timeout Preload Register A (CRC_WDTPLD2) .....	566
14.4.26	CRC Channel 2 Block Complete Timeout Preload Register B (CRC_BCTOPLD2) .....	567
14.4.27	Channel 2 PSA Signature Low Register (PSA_SIGREGL2) .....	568
14.4.28	Channel 2 PSA Signature High Register (PSA_SIGREGH2) .....	569
14.4.29	Channel 2 CRC Value Low Register (CRC_REGL2) .....	570
14.4.30	Channel 2 CRC Value High Register (CRC_REGH2) .....	571
14.4.31	Channel 2 PSA Sector Signature Low Register (PSA_SECSIGREGL2) .....	572
14.4.32	Channel 2 PSA Sector Signature High Register (PSA_SECSIGREGH2) .....	573
14.4.33	Channel 2 Raw Data Low Register (RAW_DATAREGL2) .....	574
14.4.34	Channel 2 Raw Data High Register (RAW_DATAREGH2) .....	575
14.4.35	CRC Pattern Counter Preload Register 3 (CRC_PCOUNT_REG3) .....	576
14.4.36	CRC Sector Counter Preload Register 3 (CRC_SCOUNT_REG3) .....	577
14.4.37	CRC Current Sector Register 3 (CRC_CURSEC_REG3) .....	578
14.4.38	CRC Channel 3 Watchdog Timeout Preload Register A (CRC_WDTPLD3) .....	579
14.4.39	CRC Channel 3 Block Complete Timeout Preload Register B (CRC_BCTOPLD3) .....	580
14.4.40	Channel 3 PSA Signature Low Register (PSA_SIGREGL3) .....	581
14.4.41	Channel 3 PSA Signature High Register (PSA_SIGREGH3) .....	582
14.4.42	Channel 3 CRC Value Low Register (CRC_REGL3) .....	583
14.4.43	Channel 3 CRC Value High Register (CRC_REGH3) .....	584
14.4.44	Channel 3 PSA Sector Signature Low Register (PSA_SECSIGREGL3) .....	585
14.4.45	Channel 3 PSA Sector Signature High Register (PSA_SECSIGREGH3) .....	586
14.4.46	Channel 3 Raw Data Low Register (RAW_DATAREGL3) .....	587
14.4.47	Channel 3 Raw Data High Register (RAW_DATAREGH3) .....	588
14.4.48	CRC Pattern Counter Preload Register 4 (CRC_PCOUNT_REG4) .....	589
14.4.49	CRC Sector Counter Preload Register 4 (CRC_SCOUNT_REG4) .....	590
14.4.50	CRC Current Sector Register 4 (CRC_CURSEC_REG4) .....	591
14.4.51	CRC Channel 4 Watchdog Timeout Preload Register A (CRC_WDTPLD4) .....	592
14.4.52	CRC Channel 4 Block Complete Timeout Preload Register B (CRC_BCTOPLD4) .....	593
14.4.53	Channel 4 PSA Signature Low Register (PSA_SIGREGL4) .....	594
14.4.54	Channel 4 PSA Signature High Register (PSA_SIGREGH4) .....	595



14.4.55	Channel 4 CRC Value Low Register (CRC_REGL4)	596
14.4.56	Channel 4 CRC Value High Register (CRC_REGH4)	597
14.4.57	Channel 4 PSA Sector Signature Low Register (PSA_SECSIGREGL4)	598
14.4.58	Channel 4 PSA Sector Signature High Register (PSA_SECSIGREGH4)	599
14.4.59	Channel 4 Raw Data Low Register (RAW_DATAREGL4)	600
14.4.60	Channel 4 Raw Data High Register (RAW_DATAREGH4)	601
14.4.61	Data Bus Selection Register (MCRC_TRACE_BUS_SEL)	602
<b>15</b>	<b>Vectored Interrupt Manager (VIM) Module</b>	<b>603</b>
15.1	Overview	604
15.2	Device Level Interrupt Management	605
15.2.1	Interrupt Generation at the Peripheral	605
15.2.2	Interrupt Handling at the CPU	605
15.2.3	Software Interrupt Handling Options	606
15.3	Interrupt Handling Inside VIM	607
15.3.1	VIM Interrupt Channel Mapping	608
15.3.2	VIM Input Channel Management	610
15.4	Interrupt Vector Table (VIM RAM)	612
15.4.1	Interrupt Vector Table Operation	612
15.4.2	Enabling and Controlling the VIM Parity	613
15.4.3	Interrupt Vector Table Initialization	613
15.4.4	Interrupt Vector Table Parity Testing	613
15.5	VIM Wakeup Interrupt	615
15.6	Capture Event Sources	616
15.7	Examples	617
15.7.1	Examples - Configure CPU To Receive Interrupts	617
15.7.2	Examples - Register Vector Interrupt and Index Interrupt Handling	617
15.8	Registers	619
15.8.1	Interrupt Vector Table Parity Flag Register (PARFLG)	620
15.8.2	Interrupt Vector Table Parity Control Register (PARCTL)	621
15.8.3	Address Parity Error Register (ADDERR)	622
15.8.4	Fall-Back Address Parity Error Register (FBPARERR)	623
15.8.5	VIM Offset Vector Registers	624
15.8.6	IRQ Index Offset Vector Register (IRQINDEX)	625
15.8.7	FIQ Index Offset Vector Registers (FIQINDEX)	626
15.8.8	FIQ/IRQ Program Control Registers[0] (FIRQPR[0])	627
15.8.9	Pending Interrupt Read Location Registers[0] (INTREQ[0])	628
15.8.10	Interrupt Enable Set Registers[0] (REQENASET[0])	629
15.8.11	Interrupt Enable Clear Registers[0] (REQENACLR[0])	630
15.8.12	Wake-Up Enable Set Registers[0] (WAKEENASET[0])	631
15.8.13	Wake-Up Enable Clear Registers[0] (WAKEENACLR[0])	632
15.8.14	IRQ Interrupt Vector Register (IRQVECREG)	633
15.8.15	FIQ Interrupt Vector Register (FIQVECREG)	634
15.8.16	Capture Event Register (CAPEVT)	635
15.8.17	VIM Interrupt Control Register[0:31] (CHANCTRL[0:23])	636
<b>16</b>	<b>Direct Memory Access Controller (DMA) Module</b>	<b>639</b>
16.1	Overview	640
16.1.1	Main Features	640
16.2	Module Operation	641
16.2.1	Memory Space	641
16.2.2	DMA Data Access	641
16.2.3	Addressing Modes	641
16.2.4	DMA Channel Control Packets	642
16.2.5	Priority Queue	646

16.2.6	Data Packing and Unpacking .....	648
16.2.7	DMA Request .....	650
16.2.8	Auto-Initiation .....	651
16.2.9	Interrupts .....	651
16.2.10	Debugging .....	653
16.2.11	Power Management .....	653
16.2.12	FIFO Buffer .....	653
16.2.13	Channel Chaining .....	654
16.2.14	Memory Protection .....	655
16.2.15	Parity Checking .....	656
16.2.16	Parity Testing .....	657
16.2.17	Initializing RAM with Parity .....	657
16.3	Control Registers and Control Packets .....	658
16.3.1	Global Configuration Registers .....	661
16.3.2	Channel Configuration .....	733
<b>17</b>	<b>External Memory Interface (EMIF) .....</b>	<b>743</b>
17.1	Introduction .....	744
17.1.1	Purpose of the Peripheral .....	744
17.1.2	Features .....	744
17.1.3	Functional Block Diagram .....	745
17.2	EMIF Module Architecture .....	745
17.2.1	EMIF Clock Control .....	745
17.2.2	EMIF Requests .....	745
17.2.3	EMIF Signal Descriptions .....	746
17.2.4	SDRAM Controller and Interface .....	748
17.2.5	Asynchronous Controller and Interface .....	760
17.2.6	Data Bus Parking .....	772
17.2.7	Reset and Initialization Considerations .....	773
17.2.8	Interrupt Support .....	773
17.2.9	DMA Event Support .....	774
17.2.10	EMIF Signal Multiplexing .....	774
17.2.11	Memory Map .....	774
17.2.12	Priority and Arbitration .....	774
17.2.13	System Considerations .....	776
17.2.14	Power Management .....	777
17.2.15	Emulation Considerations .....	777
17.3	Registers .....	778
17.3.1	Module ID Register (MIDR) .....	779
17.3.2	Asynchronous Wait Cycle Configuration Register (AWCC) .....	779
17.3.3	SDRAM Configuration Register (SDCR) .....	781
17.3.4	SDRAM Refresh Control Register (SDRCR) .....	783
17.3.5	Asynchronous <i>n</i> Configuration Registers (CE2CFG-CE5CFG) .....	784
17.3.6	SDRAM Timing Register (SDTIMR) .....	785
17.3.7	SDRAM Self Refresh Exit Timing Register (SDSRETR) .....	786
17.3.8	EMIF Interrupt Raw Register (INTRAW) .....	787
17.3.9	EMIF Interrupt Masked Register (INTMSK) .....	788
17.3.10	EMIF Interrupt Mask Set Register (INTMSKSET) .....	789
17.3.11	EMIF Interrupt Mask Clear Register (INTMSKCLR) .....	790
17.3.12	Page Mode Control Register (PMCR) .....	791
17.4	Example Configuration .....	792
17.4.1	Hardware Interface .....	792
17.4.2	Software Configuration .....	792
<b>18</b>	<b>Analog To Digital Converter (ADC) Module .....</b>	<b>801</b>



18.1	Overview .....	802
18.2	Introduction .....	803
18.2.1	Input Multiplexor .....	803
18.2.2	Self-Test and Calibration Cell .....	804
18.2.3	Analog-to-Digital Converter Core .....	804
18.2.4	Sequencer .....	804
18.2.5	Conversion Groups .....	805
18.3	Basic Features and Usage of the ADC .....	806
18.3.1	How to Select Between 12-bit and 10-bit Resolutions .....	806
18.3.2	How to Setup the ADCLK Speed .....	806
18.3.3	How to Setup the Input Channel Acquisition Time .....	806
18.3.4	How to Select an Input Channel for Conversion .....	806
18.3.5	How to Select Between Single Conversion Sequence or Continuous Conversions .....	806
18.3.6	How to Start a Conversion .....	807
18.3.7	How to Know When the Group Conversion is Completed .....	807
18.3.8	How Results are Stored in the Results' Memory .....	807
18.3.9	How to Read the Results from the Results' Memory .....	808
18.3.10	How to Stop a Conversion .....	811
18.3.11	Example Sequence for Basic Configuration of ADC Module .....	811
18.4	Advanced Conversion Group Configuration Options .....	813
18.4.1	Group Trigger Options .....	813
18.4.2	Single or Continuous Conversion Modes .....	814
18.4.3	Conversion Group Freeze Capability .....	814
18.4.4	Conversion Group Memory Overrun Option .....	815
18.4.5	Response on Writing Non-Zero Value to Conversion Group's Channel Select Register .....	815
18.4.6	Conversion Result Size on Reading: 8-bit, 10-bit or 12-bit .....	816
18.4.7	Option to Read Group Channel Id Along With Conversion Result .....	816
18.5	ADC Module Basic Interrupts .....	817
18.5.1	Group Conversion End Interrupt .....	817
18.5.2	Group Memory Threshold Interrupt .....	817
18.5.3	Group Memory Overrun Interrupt .....	817
18.6	ADC Module DMA Requests .....	818
18.6.1	DMA request for each conversion result written to the results' memory .....	818
18.6.2	DMA request for a fixed number of conversion results .....	818
18.7	ADC Magnitude Threshold Interrupts .....	819
18.7.1	Magnitude Threshold Interrupt Configuration .....	819
18.7.2	Magnitude Threshold Interrupt Comparison Mask Configuration .....	819
18.7.3	Magnitude Threshold Interrupt Enable / Disable Control .....	819
18.7.4	Magnitude Threshold Interrupt Flags .....	819
18.7.5	Magnitude Threshold Interrupt Offset Register .....	819
18.8	ADC Special Modes .....	820
18.8.1	ADC Error Calibration Mode .....	820
18.8.2	ADC Self-Test Mode .....	823
18.8.3	ADC Power-Down Mode .....	825
18.8.4	ADC Sample Capacitor Discharge Mode .....	826
18.9	ADC Results' RAM Special Features .....	827
18.9.1	ADC Results' RAM Auto-Initialization .....	827
18.9.2	ADC Results' RAM Test Mode .....	827
18.9.3	ADC Results' RAM Parity .....	827
18.10	ADEVT Pin General Purpose I/O Functionality .....	829
18.10.1	GPIO Functionality .....	829
18.10.2	Summary .....	829
18.11	ADC Control Registers .....	831

18.11.1	ADC Reset Control Register (ADRSTCR) .....	833
18.11.2	ADC Operating Mode Control Register (ADOPMODECR) .....	834
18.11.3	ADC Clock Control Register (ADCLOCKCR) .....	836
18.11.4	ADC Calibration Mode Control Register (ADCALCR) .....	837
18.11.5	ADC Event Group Operating Mode Control Register (ADEVMODECR) .....	838
18.11.6	ADC Group1 Operating Mode Control Register (ADG1MODECR) .....	840
18.11.7	ADC Group2 Operating Mode Control Register (ADG2MODECR) .....	842
18.11.8	ADC Event Group Trigger Source Select Register (ADEVSRC) .....	844
18.11.9	ADC Group1 Trigger Source Select Register (ADG1SRC) .....	845
18.11.10	ADC Group2 Trigger Source Select Register (ADG2SRC) .....	846
18.11.11	ADC Event Interrupt Enable Control Register (ADEVINTENA) .....	847
18.11.12	ADC Group1 Interrupt Enable Control Register (ADG1INTENA) .....	848
18.11.13	ADC Group2 Interrupt Enable Control Register (ADG2INTENA) .....	849
18.11.14	ADC Event Group Interrupt Flag Register (ADEVINTFLG) .....	850
18.11.15	ADC Group1 Interrupt Flag Register (ADG1INTFLG) .....	851
18.11.16	ADC Group2 Interrupt Flag Register (ADG2INTFLG) .....	852
18.11.17	ADC Event Group Threshold Interrupt Control Register (ADEVTHRINTCR) .....	853
18.11.18	ADC Group1 Threshold Interrupt Control Register (ADG1THRINTCR) .....	854
18.11.19	ADC Group2 Threshold Interrupt Control Register (ADG2THRINTCR) .....	855
18.11.20	ADC Event Group DMA Control Register (ADEVDMACR) .....	856
18.11.21	ADC Group1 DMA Control Register (ADG1DMACR) .....	857
18.11.22	ADC Group2 DMA Control Register (ADG2DMACR) .....	858
18.11.23	ADC Results Memory Configuration Register (ADBNDCR) .....	859
18.11.24	ADC Results Memory Size Configuration Register (ADBNDEND) .....	860
18.11.25	ADC Event Group Sampling Time Configuration Register (ADEVSAMP) .....	861
18.11.26	ADC Group1 Sampling Time Configuration Register (ADG1SAMP) .....	862
18.11.27	ADC Group2 Sampling Time Configuration Register (ADG2SAMP) .....	863
18.11.28	ADC Event Group Status Register (ADEVSR) .....	864
18.11.29	ADC Group1 Status Register (ADG1SR) .....	865
18.11.30	ADC Group2 Status Register (ADG2SR) .....	866
18.11.31	ADC Event Group Channel Select Register (ADEVSEL) .....	867
18.11.32	ADC Group1 Channel Select Register (ADG1SEL) .....	868
18.11.33	ADC Group2 Channel Select Register (ADG2SEL) .....	869
18.11.34	ADC Calibration and Error Offset Correction Register (ADCALR) .....	870
18.11.35	ADC State Machine Status Register (ADSMSTATE) .....	871
18.11.36	ADC Channel Last Conversion Value Register (ADLASTCONV) .....	872
18.11.37	ADC Event Group Results' FIFO Register (ADEVBUFFER) .....	873
18.11.38	ADC Group1 Results FIFO Register (ADG1BUFFER) .....	874
18.11.39	ADC Group2 Results FIFO Register (ADG2BUFFER) .....	875
18.11.40	ADC Event Group Results Emulation FIFO Register (ADEVEMUBUFFER) .....	876
18.11.41	ADC Group1 Results Emulation FIFO Register (ADG1EMUBUFFER) .....	876
18.11.42	ADC Group2 Results Emulation FIFO Register (ADG2EMUBUFFER) .....	876
18.11.43	ADC ADEVT Pin Direction Control Register (ADEVTDIR) .....	877
18.11.44	ADC ADEVT Pin Output Value Control Register (ADEVTOUT) .....	878
18.11.45	ADC ADEVT Pin Input Value Register (ADEVTIN) .....	879
18.11.46	ADC ADEVT Pin Set Register (ADEVTSET) .....	880
18.11.47	ADC ADEVT Pin Clear Register (ADEVTCLR) .....	881
18.11.48	ADC ADEVT Pin Open Drain Enable Register (ADEVTPDR) .....	882
18.11.49	ADC ADEVT Pin Pull Control Disable Register (ADEVTPDIS) .....	883
18.11.50	ADC ADEVT Pin Pull Control Select Register (ADEVTPSEL) .....	884
18.11.51	ADC Event Group Sample Cap Discharge Control Register (ADEVSAMPDISEN) .....	885
18.11.52	ADC Group1 Sample Cap Discharge Control Register (ADG1SAMPDISEN) .....	886
18.11.53	ADC Group2 Sample Cap Discharge Control Register (ADG2SAMPDISEN) .....	887

18.11.54	ADC Magnitude Compare Interrupt Control Registers (ADMAGINTxCR) .....	888
18.11.55	ADC Magnitude Compare Interrupt Mask Register (ADMAGINTxMASK) .....	889
18.11.56	ADC Magnitude Compare Interrupt Enable Set Register (ADMAGINTENASET) .....	890
18.11.57	ADC Magnitude Compare Interrupt Enable Clear Register (ADMAGINTENACLR) .....	891
18.11.58	ADC Magnitude Compare Interrupt Flag Register (ADMAGINTFLG) .....	892
18.11.59	ADC Magnitude Compare Interrupt Offset Register (ADMAGINTOFF) .....	893
18.11.60	ADC Event Group FIFO Reset Control Register (ADEVFIFORESETCR) .....	894
18.11.61	ADC Group1 FIFO Reset Control Register (ADG1FIFORESETCR) .....	895
18.11.62	ADC Group2 FIFO Reset Control Register (ADG2FIFORESETCR) .....	896
18.11.63	ADC Event Group RAM Write Address Register (ADEVRAMWRADDR) .....	897
18.11.64	ADC Group1 RAM Write Address Register (ADG1RAMWRADDR) .....	898
18.11.65	ADC Group2 RAM Write Address Register (ADG2RAMWRADDR) .....	899
18.11.66	ADC Parity Control Register (ADPARCR) .....	900
18.11.67	ADC Parity Error Address Register (ADPARADDR) .....	901
18.11.68	ADC Power-Up Delay Control Register (ADPWRUPDLYCTRL) .....	902
<b>19</b>	<b>High-End Timer (N2HET) Module .....</b>	<b>903</b>
19.1	Features .....	904
19.1.1	Overview .....	904
19.1.2	Block Diagram .....	906
19.2	N2HET Functional Description .....	908
19.2.1	Specialized Timer Micromachine .....	908
19.2.2	N2HET RAM Organization .....	912
19.2.3	Time Base .....	915
19.2.4	Host Interface .....	917
19.2.5	I/O Control .....	918
19.2.6	Suppression Filters .....	932
19.2.7	Interrupts and Exceptions .....	933
19.2.8	Hardware Priority Scheme: .....	934
19.2.9	N2HET Requests to DMA and HTU .....	935
19.3	Angle Functions .....	937
19.3.1	Software Angle Generator .....	937
19.4	N2HET Control Registers .....	941
19.4.1	N2HET Register Summary .....	941
19.4.2	Global Configuration Register (HETGCR) .....	942
19.4.3	Prescale Factor Register (HETPFR) .....	944
19.4.4	N2HET Current Address Register (HETADDR) .....	945
19.4.5	Offset Index Priority Level 1 Register (HETOFF1) .....	946
19.4.6	Offset Index Priority Level 2 Register (HETOFF2) .....	947
19.4.7	Interrupt Enable Set Register (HETINTENAS) .....	948
19.4.8	Interrupt Enable Clear Register (HETINTENAC) .....	949
19.4.9	Exception Control Register 1 (HETEXC1) .....	950
19.4.10	Exception Control Register 2 (HETEXC2) .....	951
19.4.11	Interrupt Priority Register (HETPRY) .....	952
19.4.12	Interrupt Flag Register (HETFLG) .....	953
19.4.13	AND Share Control Register (HETAND) .....	954
19.4.14	HR Share Control Register (HETHRSH) .....	955
19.4.15	XOR Share Control Register (HETXOR) .....	956
19.4.16	Request Enable Set Register (HETREQENS) .....	957
19.4.17	Request Enable Clear Register (HETREQENC) .....	958
19.4.18	Request Destination Select Register (HETREQDS) .....	959
19.4.19	NHET Direction Register (HETDIR) .....	960
19.4.20	N2HET Data Input Register (HETDIN) .....	961
19.4.21	N2HET Data Output Register (HETDOUT) .....	962

19.4.22	NHET Data Set Register (HETDSET)	963
19.4.23	N2HET Data Clear Register (HETDCLR)	964
19.4.24	N2HET Open Drain Register (HETPDR)	965
19.4.25	N2HET Pull Disable Register (HETPULDIS)	966
19.4.26	N2HET Pull Select Register (HETPSL)	967
19.4.27	Parity Control Register (HETPCR)	968
19.4.28	Parity Address Register (HETPAR)	969
19.4.29	Parity Pin Register (HETPPR)	970
19.4.30	Suppression Filter Preload Register (HETSPRLD)	971
19.4.31	Suppression Filter Enable Register (HETSFENA)	972
19.4.32	Loop Back Pair Select Register (HETLBPEL)	973
19.4.33	Loop Back Pair Direction Register (HETLBPDIR)	974
19.4.34	N2HET Pin Disable Register (HETPINDIS)	975
19.5	Instruction Set	976
19.5.1	Instruction Summary	976
19.5.2	Abbreviations, Encoding Formats and Bits	978
19.5.3	Instruction Description	981
<b>20</b>	<b>High End Timer Transfer Unit (HTU) Module</b>	<b>1047</b>
20.1	Overview	1048
20.1.1	Features	1048
20.2	Module Operation	1049
20.2.1	Data Transfers between Main RAM and NHET RAM	1051
20.2.2	Arbitration of HTU Elements and Frames	1056
20.2.3	Conditions for Frame Transfer Interruption	1056
20.2.4	HTU Overload and Request Lost Detection	1057
20.2.5	Memory Protection	1059
20.2.6	Control Packet RAM Parity Checking	1060
20.3	Use Cases	1062
20.3.1	Example: Single Element Transfer with One Trigger Request	1062
20.3.2	Example: Multiple Element Transfer with One Trigger Request	1062
20.3.3	Example: 64-Bit-Transfer of Control Field and Data Fields	1063
20.4	Control Registers	1065
20.4.1	Global Control Register (HTU GC)	1066
20.4.2	Control Packet Enable Register (HTU CPENA)	1067
20.4.3	Control Packet (CP) Busy Register 0 (HTU BUSY0)	1068
20.4.4	Control Packet (CP) Busy Register 1 (HTU BUSY1)	1069
20.4.5	Control Packet (CP) Busy Register 2 (HTU BUSY2)	1070
20.4.6	Control Packet (CP) Busy Register 3 (HTU BUSY3)	1071
20.4.7	Active Control Packet and Error Register (HTU ACPE)	1072
20.4.8	Request Lost and Bus Error Control Register (HTU RLBECTRL)	1074
20.4.9	Buffer Full Interrupt Enable Set Register (HTU BFINTS)	1075
20.4.10	Buffer Full Interrupt Enable Clear Register (HTU BFINTC)	1076
20.4.11	Interrupt Mapping Register (HTU INTMAP)	1077
20.4.12	Interrupt Offset Register 0 (HTU INTOFF0)	1078
20.4.13	Interrupt Offset Register 1 (HTU INTOFF1)	1080
20.4.14	Buffer Initialization Mode Register (HTU BIM)	1081
20.4.15	Request Lost Flag Register (HTU RLOSTFL)	1083
20.4.16	Buffer Full Interrupt Flag Register (HTU BFINTFL)	1084
20.4.17	BER Interrupt Flag Register (HTU BERINTFL)	1085
20.4.18	Memory Protection 1 Start Address Register (HTU MP1S)	1086
20.4.19	Memory Protection 1 End Address Register (HTU MP1E)	1087
20.4.20	Debug Control Register (HTU DCTRL)	1088
20.4.21	Watch Point Register (HTU WPR)	1089

20.4.22	Watch Mask Register (HTU WMR) .....	1090
20.4.23	Module Identification Register (HTU ID) .....	1091
20.4.24	Parity Control Register (HTU PCR) .....	1092
20.4.25	Parity Address Register (HTU PAR) .....	1093
20.4.26	Memory Protection Control and Status Register (HTU MPCS) .....	1094
20.4.27	Memory Protection Start Address Register 0 (HTU MPOS) .....	1096
20.4.28	Memory Protection End Address Register (HTU MPOE) .....	1097
20.5	Double Control Packet Configuration Memory .....	1098
20.5.1	Initial Full Address A Register (HTU IFADDRA) .....	1099
20.5.2	Initial Full Address B Register (HTU IFADDRB) .....	1100
20.5.3	Initial NHET Address and Control Register (HTU IHADDRCT) .....	1101
20.5.4	Initial Transfer Count Register (HTU ITCOUNT) .....	1103
20.5.5	Current Full Address A Register (HTU CFADDRA) .....	1104
20.5.6	Current Full Address B Register (HTU CFADDRB) .....	1105
20.5.7	Current Frame Count Register (HTU CFCOUNT) .....	1106
20.6	Examples .....	1107
20.6.1	Application Examples for Setting the Transfer Modes of CP A and B of a DCP .....	1107
20.6.2	Software Example Sequence Assuming Circular Mode for both CP A and B .....	1107
20.6.3	Example of an Interrupt Dispatch Flow for a Request Lost Interrupt .....	1108
<b>21</b>	<b>General-Purpose Input/Output (GIO) Module .....</b>	<b>1109</b>
21.1	Overview .....	1110
21.2	Quick Start Guide .....	1111
21.3	Functional Description of GIO Module .....	1113
21.3.1	GIO Block Diagram .....	1113
21.3.2	I/O Blocks .....	1114
21.3.3	External Interrupt Block .....	1116
21.4	Device Modes of Operation .....	1121
21.4.1	Emulation Mode .....	1121
21.4.2	Power-Down Mode (Low-Power Mode) .....	1121
21.5	GIO Control Registers .....	1122
21.5.1	GIO Global Control Register (GIOGCR0) .....	1123
21.5.2	GIO Interrupt Detect Register (GIOINTDET) .....	1124
21.5.3	GIO Interrupt Polarity Register (GIOPOL) .....	1125
21.5.4	GIO Interrupt Enable Registers (GIOENASET and GIOENACL) .....	1126
21.5.5	GIO Interrupt Priority Registers (GIOLVLSET and GIOLVLCLR) .....	1128
21.5.6	GIO Interrupt Flag Register (GIOFLG) .....	1130
21.5.7	GIO Offset A Register (GIOOFFA) .....	1131
21.5.8	GIO Offset B Register (GIOOFFB) .....	1132
21.5.9	GIO Emulation A Register (GIOEMUA) .....	1133
21.5.10	GIO Emulation B Register (GIOEMUB) .....	1134
21.5.11	GIO Data Direction Registers [A-B][7:0] (GIODIR[A-B][7:0]) .....	1135
21.5.12	GIO Data Input Registers [A-B][7:0] (GIODIN[A-B][7:0]) .....	1136
21.5.13	GIO Data Output Registers [A-B][7:0] (GIODOUT[A-B][7:0]) .....	1137
21.5.14	GIO Data Set Register [A-B][7:0] (GIODSET[A-B][7:0]) .....	1138
21.5.15	GIO Data Clear Registers [A-B][7:0] (GIODCLR[A-B][7:0]) .....	1139
21.5.16	GIO Open Drain Register [A-B][7:0] (GIOPDR[A-B][7:0]) .....	1140
21.5.17	GIO Pull Disable Registers [A-B][7:0] (GIOPULDIS[A-B][7:0]) .....	1141
21.5.18	GIO Pull Select Register [A-B][7:0] (GIOPSL[A-B][7:0]) .....	1142
21.6	Applications Examples .....	1143
21.6.1	Example: Setting Interrupts and Configuring Pins for Output .....	1143
21.6.2	Example: Toggling Output Buffers .....	1144
21.6.3	Example: Clearing Interrupt Flags and Setting Interrupts .....	1145
21.6.4	Example: Reading Port B Input Register .....	1146



<b>22</b>	<b>Controller Area Network (DCAN) Module</b>	<b>1147</b>
22.1	Overview	1148
22.1.1	Features	1148
22.1.2	Functional Description	1148
22.2	CAN Modules	1150
22.3	CAN Blocks	1151
22.3.1	CAN Core	1151
22.3.2	Message RAM	1151
22.3.3	Message Handler	1151
22.3.4	Message RAM Interface	1151
22.3.5	Register and Message Object Access	1152
22.3.6	Dual Clock Source	1152
22.4	CAN Bit Timing	1153
22.4.1	Bit Time and Bit Rate	1153
22.4.2	DCAN Bit Timing Registers	1155
22.5	CAN Module Configuration	1158
22.5.1	DCAN RAM Initialization through hardware	1158
22.5.2	CAN Module Initialization	1158
22.6	Message RAM	1161
22.6.1	Structure of Message Objects	1161
22.6.2	Addressing Message Objects in RAM	1163
22.6.3	Message RAM Representation in Debug/Suspend Mode	1164
22.6.4	Message RAM Representation in Direct Access Mode	1164
22.7	Message Interface Register Sets	1166
22.7.1	Message Interface Register Sets 1 and 2	1166
22.7.2	Using Message Interface Register Sets 1 and 2	1166
22.7.3	Message Interface Register 3	1167
22.8	Message Object Configurations	1169
22.8.1	Configuration of a Transmit Object for Data Frames	1169
22.8.2	Configuration of a Transmit Object for Remote Frames	1169
22.8.3	Configuration of a Single Receive Object for Data Frames	1169
22.8.4	Configuration of a Single Receive Object for Remote Frames	1170
22.8.5	Configuration of a FIFO Buffer	1170
22.8.6	Reconfiguration of Message Objects for the Reception of Frames	1170
22.8.7	Reconfiguration of Message Objects for the Reception of Frames	1170
22.9	Message Handling	1171
22.9.1	Message Handler Overview	1171
22.9.2	Receive/Transmit Priority	1171
22.9.3	Transmission of Messages in Event Driven CAN Communication	1172
22.9.4	Updating a Transmit Object	1172
22.9.5	Changing a Transmit Object	1172
22.9.6	Acceptance Filtering of Received Messages	1173
22.9.7	Reception of Data Frames	1173
22.9.8	Reception of Remote Frames	1173
22.9.9	Reading Received Messages	1173
22.9.10	Requesting New Data for a Receive Object	1174
22.9.11	Storing Received Messages in FIFO Buffers	1174
22.9.12	Reading from a FIFO Buffer	1174
22.10	CAN Message Transfer	1176
22.10.1	Automatic Retransmission	1176
22.10.2	Auto-Bus-On	1176
22.11	Interrupt Functionality	1178
22.11.1	Message Object Interrupts	1178

22.11.2	Status Change Interrupts .....	1178
22.11.3	Error Interrupts .....	1178
22.12	Global Power Down Mode .....	1180
22.12.1	Entering Global Power Down Mode .....	1180
22.12.2	Wakeup From Global Power Down Mode .....	1180
22.13	Local Power Down Mode .....	1181
22.13.1	Entering Local Power Down Mode .....	1181
22.13.2	Wakeup From Local Power Down .....	1181
22.14	GIO support .....	1183
22.15	Test Modes .....	1184
22.15.1	Silent Mode .....	1184
22.15.2	Loop Back Mode .....	1184
22.15.3	External Loop Back Mode .....	1185
22.15.4	Loop Back Combined with Silent Mode .....	1186
22.15.5	Software Control of CAN_TX pin .....	1187
22.16	Parity Check Mechanism .....	1188
22.16.1	Behavior on Parity Error .....	1188
22.16.2	Parity Testing .....	1188
22.17	Debug/Suspend Mode .....	1189
22.18	DCAN Control Registers .....	1190
22.18.1	CAN Control Register (DCAN CTL) .....	1192
22.18.2	Error and Status Register (DCAN ES) .....	1195
22.18.3	Error Counter Register (DCAN ERRC) .....	1197
22.18.4	Bit Timing Register (DCAN BTR) .....	1198
22.18.5	Interrupt Register (DCAN INT) .....	1199
22.18.6	Test Register (DCAN TEST) .....	1200
22.18.7	Parity Error Code Register (DCAN PERR) .....	1201
22.18.8	Auto-Bus-On Time Register (DCAN ABOTR) .....	1202
22.18.9	Transmission Request X Register (DCAN TXRQ X) .....	1203
22.18.10	Transmission Request Registers (DCAN TXRQ12 to DCAN TXRQ78) .....	1204
22.18.11	New Data X Register (DCAN NWDAT X) .....	1205
22.18.12	New Data Registers (DCAN NWDAT12 to DCAN NWDAT78) .....	1206
22.18.13	Interrupt Pending X Register (DCAN INTPND X) .....	1207
22.18.14	Interrupt Pending Registers (DCAN INTPND12 to DCAN INTPND78) .....	1208
22.18.15	Message Valid X Register (DCAN MSGVAL X) .....	1209
22.18.16	Message Valid Registers (DCAN MSGVAL12 to DCAN MSGVAL78) .....	1210
22.18.17	Interrupt Multiplexer Registers (DCAN INTMUX12 to DCAN INTMUX78) .....	1211
22.18.18	IF1/IF2 Command Registers (DCAN IF1CMD, DCAN IF2CMD) .....	1212
22.18.19	IF1/IF2 Mask Registers (DCAN IF1MSK, DCAN IF2MSK) .....	1215
22.18.20	IF1/IF2 Arbitration Registers (DCAN IF1ARB, DCAN IF2ARB) .....	1216
22.18.21	IF1/IF2 Message Control Registers (DCAN IF1MCTL, DCAN IF2MCTL) .....	1218
22.18.22	IF1/IF2 Data A and Data B Registers (DCAN IF1DATA/DATB, DCAN IF2DATA/DATB) .....	1220
22.18.23	IF3 Observation Register (DCAN IF3OBS) .....	1221
22.18.24	IF3 Mask Register (DCAN IF3MSK) .....	1223
22.18.25	IF3 Arbitration Register (DCAN IF3ARB) .....	1224
22.18.26	IF3 Message Control Register (DCAN IF3MCTL) .....	1225
22.18.27	IF3 Data A and Data B Registers (DCAN IF3DATA/DATB) .....	1227
22.18.28	IF3 Update Enable Registers (DCAN IF3UPD12 to IF3UPD78) .....	1228
22.18.29	CAN TX IO Control Register (DCAN TIOC) .....	1229
22.18.30	CAN RX IO Control Register (DCAN RIOC) .....	1231
<b>23</b>	<b>Multi-Buffered Serial Peripheral Interface Module (MibSPI) with Parallel Pin Option (MibSPIP)</b>	<b>1233</b>
23.1	Overview .....	1234
23.1.1	Word Format Options .....	1234

23.1.2	Multi-buffering (Mib) support .....	1234
23.1.3	Transmission Lock (Multi-Buffer Mode Master Only) .....	1235
23.2	Operating Modes .....	1236
23.2.1	Pin Configurations .....	1236
23.2.2	Data Handling .....	1236
23.2.3	Operation with SPISCS .....	1238
23.2.4	Operation with SPIENA .....	1239
23.2.5	Five-Pin Operation (Hardware Handshaking) .....	1240
23.2.6	Data Formats .....	1241
23.2.7	Clocking Modes .....	1242
23.2.8	Data Transfer Example .....	1244
23.2.9	Decoded and Encoded Chip Select (Master Only) .....	1245
23.2.10	Variable Chip Select Setup and Hold Timing (Master Only) .....	1245
23.2.11	Hold Chip-Select Active .....	1245
23.2.12	Detection of Slave Desynchronization (Master Only) .....	1246
23.2.13	ENA Signal Time-Out (Master Only) .....	1246
23.2.14	Data-Length Error .....	1247
23.2.15	Parallel Mode (Multiple SIMO/SOMI Support, not available on all devices) .....	1247
23.2.16	Continuous Self-Test (Master/Slave) .....	1254
23.3	Test Features .....	1255
23.3.1	Internal Loop-Back Test Mode (Master Only) .....	1255
23.3.2	Input/Output Loopback Test Mode .....	1255
23.4	General-Purpose I/O .....	1257
23.5	Low-Power Mode .....	1258
23.6	Interrupts .....	1259
23.6.1	Interrupts in Multi-Buffer Mode .....	1259
23.7	DMA Interface .....	1261
23.7.1	DMA in Multi-Buffer Mode .....	1261
23.8	Module Configuration .....	1262
23.8.1	Compatibility(SPI) Mode Configuration .....	1262
23.8.2	MibSPI Mode Configuration .....	1262
23.9	Control Registers .....	1264
23.9.1	SPI Global Control Register 0 (SPIGCR0) .....	1266
23.9.2	SPI Global Control Register 1 (SPIGCR1) .....	1267
23.9.3	SPI Interrupt Register (SPIINT0) .....	1269
23.9.4	SPI Interrupt Level Register (SPILVL) .....	1271
23.9.5	SPI Flag Register (SPIFLG) .....	1272
23.9.6	SPI Pin Control Register 0 (SPIPC0) .....	1275
23.9.7	SPI Pin Control Register 1 (SPIPC1) .....	1277
23.9.8	SPI Pin Control Register 2 (SPIPC2) .....	1279
23.9.9	SPI Pin Control Register 3 (SPIPC3) .....	1280
23.9.10	SPI Pin Control Register 4 (SPIPC4) .....	1282
23.9.11	SPI Pin Control Register 5 (SPIPC5) .....	1284
23.9.12	SPI Pin Control Register 6 (SPIPC6) .....	1286
23.9.13	SPI Pin Control Register 7 (SPIPC7) .....	1288
23.9.14	SPI Pin Control Register 8 (SPIPC8) .....	1290
23.9.15	SPI Transmit Data Register 0 (SPIDAT0) .....	1292
23.9.16	SPI Transmit Data Register 1 (SPIDAT1) .....	1293
23.9.17	SPI Receive Buffer Register (SPIBUF) .....	1294
23.9.18	SPI Emulation Register (SPIEMU) .....	1296
23.9.19	SPI Delay Register (SPIDELAY) .....	1297
23.9.20	SPI Default Chip Select Register (SPIDEF) .....	1300
23.9.21	SPI Data Format Registers (SPIFMT) .....	1301



23.9.22	Interrupt Vector 0 (INTVECT0) .....	1303
23.9.23	Interrupt Vector 1 (INTVECT1) .....	1305
23.9.24	Parallel/Modulo Mode Control Register (SPIPMCTRL) .....	1307
23.9.25	Multi-buffer Mode Enable Register (MIBSPIE) .....	1310
23.9.26	TG Interrupt Enable Set Register (TGITENST) .....	1311
23.9.27	TG Interrupt Enable Clear Register (TGITENCR) .....	1312
23.9.28	Transfer Group Interrupt Level Set Register (TGITLVST) .....	1313
23.9.29	Transfer Group Interrupt Level Clear Register (TGITLVCR) .....	1314
23.9.30	Transfer Group Interrupt Flag Register (TGINTFLAG) .....	1315
23.9.31	Tick Count Register (TICKCNT) .....	1316
23.9.32	Last TG End Pointer (LTGPEND) .....	1317
23.9.33	TGx Control Registers (TGxCTRL) .....	1318
23.9.34	DMA Channel Control Register (DMAxCTRL) .....	1321
23.9.35	DMAxCOUNT Register (ICOUNT) .....	1323
23.9.36	DMA Large Count (DMACNTLEN) .....	1324
23.9.37	Multi-buffer RAM Uncorrectable Parity Error Control Register (UERRCTRL) .....	1325
23.9.38	Multi-buffer RAM Uncorrectable Parity Error Status Register (UERRSTAT) .....	1326
23.9.39	RXRAM Uncorrectable Parity Error Address Register (UERRADDR1) .....	1327
23.9.40	TXRAM Uncorrectable Parity Error Address Register (UERRADDR0) .....	1328
23.9.41	RXRAM Overrun Buffer Address Register (RXOVRN_BUF_ADDR) .....	1329
23.9.42	I/O-Loopback Test Control Register (IOLPBKTSTCR) .....	1330
23.10	Multi-Buffer RAM .....	1332
23.10.1	Multi-Buffer RAM Auto Initialization .....	1332
23.10.2	Multi-buffer RAM Register Summary .....	1334
23.10.3	Multi-buffer RAM Transmit Data Register .....	1335
23.10.4	Multi-buffer RAM Receive Buffer Register .....	1337
23.11	Parity Memory .....	1339
23.11.1	Example of Parity Memory Organization .....	1340
23.12	MibSPI Pin Timing Parameters .....	1342
23.12.1	Master Mode Timings for SPI/MibSPI .....	1342
23.12.2	Slave Mode Timings for SPI/MibSPI .....	1343
23.12.3	Master Mode Timing Parameter details .....	1344
23.12.4	Slave Mode Timing Parameter details .....	1346
<b>24</b>	<b>Serial Communication Interface (SCI)/Local Interconnect Network (LIN) Module .....</b>	<b>1347</b>
24.1	Introduction and Features .....	1348
24.1.1	SCI Features .....	1348
24.1.2	LIN Features .....	1348
24.1.3	Block Diagram .....	1349
24.2	SCI Communication Formats .....	1353
24.2.1	SCI Frame Formats .....	1353
24.2.2	SCI Timing Mode .....	1353
24.2.3	SCI Baud Rate .....	1354
24.2.4	SCI Multiprocessor Communication Modes .....	1356
24.2.5	SCI Multi Buffered Mode .....	1358
24.3	SCI Interrupts .....	1361
24.3.1	Transmit Interrupt .....	1362
24.3.2	Receive Interrupt .....	1362
24.3.3	WakeUp Interrupt .....	1362
24.3.4	Error Interrupts .....	1363
24.4	SCI DMA Interface .....	1364
24.4.1	Receive DMA Requests .....	1364
24.4.2	Transmit DMA Requests .....	1364
24.5	SCI Configurations .....	1365

24.5.1	Receiving Data .....	1365
24.5.2	Transmitting Data .....	1366
24.6	SCI Low Power Mode .....	1367
24.6.1	Sleep Mode for Multiprocessor Communication .....	1367
24.7	LIN Communication Formats .....	1369
24.7.1	LIN Standards .....	1369
24.7.2	Message Frame .....	1369
24.7.3	Synchronizer .....	1371
24.7.4	Baud Rate .....	1371
24.7.5	Header Generation .....	1373
24.7.6	Extended Frames Handling .....	1377
24.7.7	Timeout Control .....	1378
24.7.8	TXRX Error Detector (TED) .....	1379
24.7.9	Message Filtering and Validation .....	1381
24.7.10	Receive Buffers .....	1383
24.7.11	Transmit Buffers .....	1384
24.8	LIN Interrupts .....	1385
24.9	LIN DMA Interface .....	1386
24.9.1	LIN Receive DMA Requests .....	1386
24.9.2	LIN Transmit DMA Requests .....	1386
24.10	LIN Configurations .....	1387
24.10.1	Receiving Data .....	1387
24.10.2	Transmitting Data .....	1388
24.11	Low-Power Mode .....	1389
24.11.1	Entering Sleep Mode .....	1389
24.11.2	Wakeup .....	1389
24.11.3	Wakeup Timeouts .....	1390
24.12	Emulation Mode .....	1391
24.13	SCI/LIN Control Registers .....	1392
24.13.1	SCI Global Control Register 0 (SCIGCR0) .....	1393
24.13.2	SCI Global Control Register 1 (SCIGCR1) .....	1394
24.13.3	SCI Global Control Register 2 (SCIGCR2) .....	1398
24.13.4	SCI Set Interrupt Register (SCISSETINT) .....	1400
24.13.5	SCI Clear Interrupt Register (SCICLEARINT) .....	1403
24.13.6	SCI Set Interrupt Level Register (SCISSETINTLVL) .....	1407
24.13.7	SCI Clear Interrupt Level Register (SCICLEARINTLVL) .....	1410
24.13.8	SCI Flags Register (SCIFLR) .....	1413
24.13.9	SCI Interrupt Vector Offset 0 (SCIINTVECT0) .....	1420
24.13.10	SCI Interrupt Vector Offset 1 (SCIINTVECT1) .....	1421
24.13.11	SCI Format Control Register (SCIFORMAT) .....	1422
24.13.12	Baud Rate Selection Register (BRS) .....	1423
24.13.13	SCI Data Buffers (SCIED, SCIRD, SCITD) .....	1425
24.13.14	SCI Pin I/O Control Register 0 (SCIPIO0) .....	1428
24.13.15	SCI Pin I/O Control Register 1 (SCIPIO1) .....	1429
24.13.16	SCI Pin I/O Control Register 2 (SCIPIO2) .....	1430
24.13.17	SCI Pin I/O Control Register 3 (SCIPIO3) .....	1431
24.13.18	SCI Pin I/O Control Register 4 (SCIPIO4) .....	1432
24.13.19	SCI Pin I/O Control Register 5 (SCIPIO5) .....	1433
24.13.20	SCI Pin I/O Control Register 6 (SCIPIO6) .....	1434
24.13.21	SCI Pin I/O Control Register 7 (SCIPIO7) .....	1435
24.13.22	SCI Pin I/O Control Register 8 (SCIPIO8) .....	1436
24.13.23	LIN Compare Register (LINCOMPARE) .....	1437
24.13.24	LIN Receive Buffer 0 Register (LINRD0) .....	1438

24.13.25	LIN Receive Buffer 1 Register (LINRD1) .....	1439
24.13.26	LIN Mask Register (LINMASK) .....	1440
24.13.27	LIN Identification Register (LINID) .....	1441
24.13.28	LIN Transmit Buffer 0 Register(LINTD0) .....	1442
24.13.29	LIN Transmit Buffer 1 Register (LINTD1) .....	1443
24.13.30	Maximum Baud Rate Selection Register (MBRS) .....	1444
24.13.31	Input/Output Error Enable (IODFTCTRL) Register .....	1445
24.14	GPIO Functionality .....	1447
24.14.1	GPIO Functionality .....	1447
24.14.2	Under Reset .....	1447
24.14.3	Out of Reset .....	1447
24.14.4	Open-Drain Feature Enabled on a Pin .....	1448
24.14.5	Summary .....	1448
<b>25</b>	<b>Serial Communication Interface (SCI) Module .....</b>	<b>1449</b>
25.1	Introduction .....	1450
25.1.1	SCI Features .....	1450
25.1.2	Block Diagram .....	1450
25.2	SCI Communication Formats .....	1452
25.2.1	SCI Frame Formats .....	1452
25.2.2	SCI Timing Mode .....	1452
25.2.3	SCI Baud Rate .....	1453
25.2.4	SCI Multiprocessor Communication Modes .....	1454
25.3	SCI Interrupts .....	1457
25.3.1	Transmit Interrupt .....	1458
25.3.2	Receive Interrupt .....	1458
25.3.3	WakeUp Interrupt .....	1458
25.3.4	Error Interrupts .....	1459
25.4	SCI DMA Interface .....	1460
25.4.1	Receive DMA Requests .....	1460
25.4.2	Transmit DMA Requests .....	1460
25.5	SCI Configurations .....	1461
25.5.1	Receiving Data .....	1461
25.5.2	Transmitting Data .....	1461
25.6	SCI Low Power Mode .....	1463
25.6.1	Sleep Mode for Multiprocessor Communication .....	1463
25.7	SCI Control Registers .....	1465
25.7.1	SCI Global Control Register 0 (SCIGCR0) .....	1466
25.7.2	SCI Global Control Register 1 (SCIGCR1) .....	1467
25.7.3	SCI Set Interrupt Register (SCISSETINT) .....	1470
25.7.4	SCI Clear Interrupt Register (SCICLEARINT) .....	1472
25.7.5	SCI Set Interrupt Level Register (SCISSETINTLVL) .....	1474
25.7.6	SCI Clear Interrupt Level Register (SCICLEARINTLVL) .....	1476
25.7.7	SCI Flags Register (SCIFLR) .....	1478
25.7.8	SCI Interrupt Vector Offset 0 (SCIINTVECT0) .....	1482
25.7.9	SCI Interrupt Vector Offset 1 (SCIINTVECT1) .....	1483
25.7.10	SCI Format Control Register (SCIFORMAT) .....	1484
25.7.11	Baud Rate Selection Register (BRS) .....	1485
25.7.12	SCI Data Buffers (SCIED, SCIRD, SCITD) .....	1487
25.7.13	SCI Pin I/O Control Register 0 (SCIPIO0) .....	1490
25.7.14	SCI Pin I/O Control Register 1 (SCIPIO1) .....	1491
25.7.15	SCI Pin I/O Control Register 2 (SCIPIO2) .....	1492
25.7.16	SCI Pin I/O Control Register 3 (SCIPIO3) .....	1493
25.7.17	SCI Pin I/O Control Register 4 (SCIPIO4) .....	1494

25.7.18	SCI Pin I/O Control Register 5 (SCIPIO5)	1495
25.7.19	SCI Pin I/O Control Register 6 (SCIPIO6)	1496
25.7.20	SCI Pin I/O Control Register 7 (SCIPIO7)	1497
25.7.21	SCI Pin I/O Control Register 8 (SCIPIO8)	1498
25.7.22	Input/Output Error Enable (IODFTCTRL) Register	1499
25.8	GPIO Functionality	1501
25.8.1	GPIO Functionality	1501
25.8.2	Under Reset	1501
25.8.3	Out of Reset	1501
25.8.4	Open-Drain Feature Enabled on a Pin	1502
25.8.5	Summary	1502
<b>26</b>	<b>Inter-Integrated Circuit (I2C) Module</b>	<b>1503</b>
26.1	Overview	1504
26.1.1	Introduction to the I2C Module	1504
26.1.2	Functional Overview	1505
26.1.3	Clock Generation	1507
26.2	I2C Module Operation	1509
26.2.1	Input and Output Voltage Levels	1509
26.2.2	I2C Module Reset Conditions	1509
26.2.3	I2C Module Data Validity	1509
26.2.4	I2C Module Start and Stop Conditions	1509
26.2.5	Serial Data Formats	1510
26.2.6	NACK Bit Generation	1511
26.3	I2C Operation Modes	1513
26.3.1	Master Transmitter Mode	1513
26.3.2	Master Receiver Mode	1513
26.3.3	Slave Transmitter Mode	1513
26.3.4	Slave Receiver Mode	1513
26.3.5	Low Power Mode	1514
26.3.6	Free Run Mode	1514
26.3.7	Ignore NACK Mode	1514
26.4	I2C Module Integrity	1515
26.4.1	Arbitration	1515
26.4.2	I2C Clock Generation and Synchronization	1515
26.4.3	Prescaler	1516
26.4.4	Noise Filter	1516
26.5	Operational Information	1517
26.5.1	I2C Module Interrupts	1517
26.5.2	DMA Controller Events	1518
26.5.3	I2C Enable/Disable	1518
26.5.4	General Purpose I/O	1518
26.5.5	Pull Up/Pull Down Function	1518
26.5.6	Open Drain Function	1519
26.6	I2C Registers	1520
26.6.1	I2C Own Address Manager (I2COAR)	1521
26.6.2	I2C Interrupt Mask Register (I2CIMR)	1522
26.6.3	I2C Status Register (I2CSTR)	1523
26.6.4	I2C Clock Divider Low Register (I2CCKL)	1526
26.6.5	I2C Clock Control High Register (I2CCKH)	1527
26.6.6	I2C Data Count Register (I2CCNT)	1528
26.6.7	I2C Data Receive Register (I2CDRR)	1529
26.6.8	I2C Slave Address Register (I2CSAR)	1530
26.6.9	I2C Data Transmit Register (I2CDXR)	1531

26.6.10	I2C Mode Register (I2CMDR) .....	1532
26.6.11	I2C Interrupt Vector Register (I2CIVR) .....	1535
26.6.12	I2C Extended Mode Register (I2CEMDR) .....	1536
26.6.13	I2C Prescale Register (I2CPSC) .....	1537
26.6.14	I2C Peripheral ID Register 1 (I2CPID1) .....	1538
26.6.15	I2C Peripheral ID Register 2 (I2CPID2) .....	1539
26.6.16	I2C DMA Control Register (I2CDMACR) .....	1540
26.6.17	I2C Pin Function Register (I2CPFNC) .....	1541
26.6.18	I2C Pin Direction Register (I2CPDIR) .....	1542
26.6.19	I2C Data Input Register (I2CDIN) .....	1543
26.6.20	I2C Data Output Register (I2CDOUR) .....	1544
26.6.21	I2C Data Set Register (I2CDSET) .....	1545
26.6.22	I2C Data Clear Register (I2CDCLR) .....	1546
26.6.23	I2C Pin Open Drain Register (I2CPDR) .....	1547
26.6.24	I2C Pull Disable Register (I2CPDIS) .....	1548
26.6.25	I2C Pull Select Register (I2CPSEL) .....	1549
26.6.26	I2C Pins Slew Rate Select Register (I2CSRS) .....	1550
26.7	Sample Waveforms .....	1551
<b>27</b>	<b>EMAC/MDIO Module .....</b>	<b>1553</b>
27.1	Introduction .....	1554
27.1.1	Purpose of the Peripheral .....	1554
27.1.2	Features .....	1554
27.1.3	Functional Block Diagram .....	1555
27.1.4	Industry Standard(s) Compliance Statement .....	1556
27.2	Architecture .....	1557
27.2.1	Clock Control .....	1557
27.2.2	Memory Map .....	1557
27.2.3	Signal Descriptions .....	1557
27.2.4	MII / RMI Signal Multiplexing Control .....	1560
27.2.5	Ethernet Protocol Overview .....	1560
27.2.6	Programming Interface .....	1562
27.2.7	EMAC Control Module .....	1573
27.2.8	MDIO Module .....	1574
27.2.9	EMAC Module .....	1579
27.2.10	MAC Interface .....	1581
27.2.11	Packet Receive Operation .....	1585
27.2.12	Packet Transmit Operation .....	1590
27.2.13	Receive and Transmit Latency .....	1591
27.2.14	Transfer Node Priority .....	1591
27.2.15	Reset Considerations .....	1592
27.2.16	Initialization .....	1593
27.2.17	Interrupt Support .....	1595
27.2.18	Power Management .....	1599
27.2.19	Emulation Considerations .....	1599
27.3	EMAC Control Module Registers .....	1600
27.3.1	EMAC Control Module Revision ID Register (REVID) .....	1601
27.3.2	EMAC Control Module Software Reset Register (SOFTRESET) .....	1602
27.3.3	EMAC Control Module Interrupt Control Register (INTCONTROL) .....	1603
27.3.4	EMAC Control Module Receive Threshold Interrupt Enable Registers (CORXTHRESHEN) .....	1604
27.3.5	EMAC Control Module Receive Interrupt Enable Registers (CORXEN) .....	1605
27.3.6	EMAC Control Module Transmit Interrupt Enable Registers (COTXEN) .....	1606
27.3.7	EMAC Control Module Miscellaneous Interrupt Enable Registers (COMISCEN) .....	1607
27.3.8	EMAC Control Module Receive Threshold Interrupt Status Registers (CORXTHRESHSTAT) .....	1608

27.3.9	EMAC Control Module Receive Interrupt Status Registers (C0RXSTAT)	1609
27.3.10	EMAC Control Module Transmit Interrupt Status Registers (C0TXSTAT)	1610
27.3.11	EMAC Control Module Miscellaneous Interrupt Status Registers (C0MISCSTAT)	1611
27.3.12	EMAC Control Module Receive Interrupts Per Millisecond Registers (C0RXIMAX)	1612
27.3.13	EMAC Control Module Transmit Interrupts Per Millisecond Registers (C0TXIMAX)	1613
27.4	MDIO Registers	1614
27.4.1	MDIO Revision ID Register (REVID)	1615
27.4.2	MDIO Control Register (CONTROL)	1616
27.4.3	PHY Acknowledge Status Register (ALIVE)	1617
27.4.4	PHY Link Status Register (LINK)	1618
27.4.5	MDIO Link Status Change Interrupt (Unmasked) Register (LINKINTRAW)	1619
27.4.6	MDIO Link Status Change Interrupt (Masked) Register (LINKINTMASKED)	1620
27.4.7	MDIO User Command Complete Interrupt (Unmasked) Register (USERINTRAW)	1621
27.4.8	MDIO User Command Complete Interrupt (Masked) Register (USERINTMASKED)	1622
27.4.9	MDIO User Command Complete Interrupt Mask Set Register (USERINTMASKSET)	1623
27.4.10	MDIO User Command Complete Interrupt Mask Clear Register (USERINTMASKCLEAR)	1624
27.4.11	MDIO User Access Register 0 (USERACCESS0)	1625
27.4.12	MDIO User PHY Select Register 0 (USERPHYSEL0)	1626
27.4.13	MDIO User Access Register 1 (USERACCESS1)	1627
27.4.14	MDIO User PHY Select Register 1 (USERPHYSEL1)	1628
27.5	EMAC Module Registers	1629
27.5.1	Transmit Revision ID Register (TXREVID)	1633
27.5.2	Transmit Control Register (TXCONTROL)	1634
27.5.3	Transmit Teardown Register (TXTEARDOWN)	1635
27.5.4	Receive Revision ID Register (RXREVID)	1636
27.5.5	Receive Control Register (RXCONTROL)	1637
27.5.6	Receive Teardown Register (RXTEARDOWN)	1638
27.5.7	Transmit Interrupt Status (Unmasked) Register (TXINTSTATRAW)	1639
27.5.8	Transmit Interrupt Status (Masked) Register (TXINTSTATMASKED)	1640
27.5.9	Transmit Interrupt Mask Set Register (TXINTMASKSET)	1641
27.5.10	Transmit Interrupt Mask Clear Register (TXINTMASKCLEAR)	1642
27.5.11	MAC Input Vector Register (MACINVECTOR)	1643
27.5.12	MAC End Of Interrupt Vector Register (MACEOIVECTOR)	1644
27.5.13	Receive Interrupt Status (Unmasked) Register (RXINTSTATRAW)	1645
27.5.14	Receive Interrupt Status (Masked) Register (RXINTSTATMASKED)	1646
27.5.15	Receive Interrupt Mask Set Register (RXINTMASKSET)	1647
27.5.16	Receive Interrupt Mask Clear Register (RXINTMASKCLEAR)	1648
27.5.17	MAC Interrupt Status (Unmasked) Register (MACINTSTATRAW)	1649
27.5.18	MAC Interrupt Status (Masked) Register (MACINTSTATMASKED)	1650
27.5.19	MAC Interrupt Mask Set Register (MACINTMASKSET)	1651
27.5.20	MAC Interrupt Mask Clear Register (MACINTMASKCLEAR)	1652
27.5.21	Receive Multicast/Broadcast/Promiscuous Channel Enable Register (RXMBPENABLE)	1653
27.5.22	Receive Unicast Enable Set Register (RXUNICASTSET)	1656
27.5.23	Receive Unicast Clear Register (RXUNICASTCLEAR)	1657
27.5.24	Receive Maximum Length Register (RXMAXLEN)	1658
27.5.25	Receive Buffer Offset Register (RXBUFFEROFFSET)	1659
27.5.26	Receive Filter Low Priority Frame Threshold Register (RXFILTERLOWTHRESH)	1660
27.5.27	Receive Channel Flow Control Threshold Registers (RX0FLOWTHRESH-RX7FLOWTHRESH)	1661
27.5.28	Receive Channel Free Buffer Count Registers (RX0FREEBUFFER-RX7FREEBUFFER)	1662
27.5.29	MAC Control Register (MACCONTROL)	1663
27.5.30	MAC Status Register (MACSTATUS)	1665
27.5.31	Emulation Control Register (EMCONTROL)	1667
27.5.32	FIFO Control Register (FIFOCONTROL)	1668



27.5.33	MAC Configuration Register (MACCONFIG) .....	1669
27.5.34	Soft Reset Register (SOFTRESET) .....	1670
27.5.35	MAC Source Address Low Bytes Register (MACSRCADDRLO) .....	1671
27.5.36	MAC Source Address High Bytes Register (MACSRCADDRHI) .....	1672
27.5.37	MAC Hash Address Register 1 (MACHASH1) .....	1673
27.5.38	MAC Hash Address Register 2 (MACHASH2) .....	1674
27.5.39	Back Off Test Register (BOFFTEST) .....	1675
27.5.40	Transmit Pacing Algorithm Test Register (TPACETEST) .....	1676
27.5.41	Receive Pause Timer Register (RXPAUSE) .....	1677
27.5.42	Transmit Pause Timer Register (TXPAUSE) .....	1678
27.5.43	MAC Address Low Bytes Register (MACADDRLO) .....	1679
27.5.44	MAC Address High Bytes Register (MACADDRHI) .....	1680
27.5.45	MAC Index Register (MACINDEX) .....	1681
27.5.46	Transmit Channel DMA Head Descriptor Pointer Registers (TX0HDP-TX7HDP) .....	1682
27.5.47	Receive Channel DMA Head Descriptor Pointer Registers (RX0HDP-RX7HDP) .....	1683
27.5.48	Transmit Channel Completion Pointer Registers (TX0CP-TX7CP) .....	1684
27.5.49	Receive Channel Completion Pointer Registers (RX0CP-RX7CP) .....	1685
27.5.50	Network Statistics Registers .....	1686
<b>28</b>	<b>Universal Serial Bus (USB) .....</b>	<b>1695</b>
28.1	Overview .....	1696
28.2	USB Host Controller .....	1697
28.2.1	USB Open Host Controller Interface Functionality .....	1697
28.2.2	USB Host Controller Differences From OHCI Specification for USB .....	1697
28.2.3	Implementation of <i>OHCI Specification for USB</i> .....	1698
28.2.4	USB Host Controller Registers .....	1699
28.2.5	USB Host Controller Reserved Registers and Reserved Bit Fields .....	1720
28.2.6	USB Host Controller Registers, USB Reset, and USB Clocking .....	1721
28.2.7	OHCI Interrupts .....	1721
28.2.8	USB Host Controller Access to System Memory .....	1722
28.2.9	Physical Addressing .....	1722
28.2.10	USB Host Controller Bus Addressing and OHCI Data Structure Pointers .....	1722
28.2.11	NULL Pointers .....	1722
28.2.12	USB Host Controller Power Management .....	1722
28.3	USB Device Controller .....	1723
28.3.1	USB Device Controller Registers .....	1723
28.3.2	USB Device Transactions .....	1748
28.3.3	Non-Isochronous, Non-Setup OUT (USB HOST→CPU) Transactions .....	1749
28.3.4	Non-Isochronous IN (CPU→USB HOST) Transactions .....	1752
28.3.5	Isochronous OUT (USB HOST→CPU) Transactions .....	1755
28.3.6	Isochronous IN (CPU→USB HOST) Transactions .....	1757
28.3.7	Control Transfers on Endpoint 0 .....	1758
28.3.8	USB Device Initialization .....	1767
28.3.9	Preparing for Transfers .....	1769
28.3.10	USB Device Interrupt Service Routine (ISR) Flowcharts .....	1771
28.3.11	Important Note on USB Device Interrupts .....	1772
28.3.12	Parsing General USB Device Interrupt .....	1772
28.3.13	Setup Interrupt Handler .....	1773
28.3.14	Endpoint 0 RX Interrupt Handler .....	1775
28.3.15	Endpoint 0 TX Interrupt Handler .....	1777
28.3.16	Device States Changed Handler .....	1779
28.3.17	Device States Attached/Unattached Handler .....	1782
28.3.18	Device State Configuration Changed Handler .....	1783
28.3.19	Device State Address Changed Handler .....	1784

28.3.20	USB Device Reset Interrupt Handler .....	1785
28.3.21	Suspend/Resume Interrupt Handler .....	1786
28.3.22	Parsing Non-ISO Endpoint-Specific Interrupt .....	1787
28.3.23	Non-ISO, Non-Control OUT Endpoint Receive Interrupt Handler .....	1788
28.3.24	Non-ISO, Non-Control IN Endpoint Transmit Interrupt Handler .....	1790
28.3.25	SOF Interrupt Handler .....	1792
28.3.26	Summary of USB Device Controller Interrupts .....	1795
28.3.27	DMA Operation .....	1796
28.3.28	Power Management .....	1805
28.4	USB Connectivity .....	1808
28.4.1	Transceiver Interface .....	1808
28.4.2	Selecting and Configuring USB Connectivity .....	1808
28.4.3	Transceiver Signaling .....	1808
28.4.4	Host Controller Connectivity With USB Transceivers .....	1809
28.4.5	USB Function Controller Connectivity With USB Transceivers .....	1810
28.4.6	USB Hardware Considerations .....	1811
<b>29</b>	<b>Data Modification Module (DMM) .....</b>	<b>1813</b>
29.1	Overview .....	1814
29.1.1	Features .....	1814
29.1.2	Block Diagram .....	1814
29.2	Module Operation .....	1815
29.2.1	Data Format .....	1815
29.2.2	Data Port .....	1817
29.2.3	Error Handling .....	1818
29.2.4	Interrupts .....	1819
29.3	Control Registers Summary .....	1820
29.3.1	DMM Global Control Register (DMMGLBCTRL) .....	1821
29.3.2	DMM Interrupt Set Register (DMMINTSET) .....	1823
29.3.3	DMM Interrupt Clear Register (DMMINTCLR) .....	1827
29.3.4	DMM Interrupt Level Register (DMMINTLVL) .....	1832
29.3.5	DMM Interrupt Flag Register (DMMINTFLG) .....	1834
29.3.6	DMM Interrupt Offset 1 Register (DMMOFF1) .....	1838
29.3.7	DMM Interrupt Offset 2 Register (DMMOFF2) .....	1839
29.3.8	DMM Direct Data Mode Destination Register (DMMDDMDEST) .....	1840
29.3.9	DMM Direct Data Mode Blocksize Register (DMMDDMBL) .....	1841
29.3.10	DMM Direct Data Mode Pointer Register (DMMDDMPT) .....	1842
29.3.11	DMM Direct Data Mode Interrupt Pointer Register (DMMINTPT) .....	1843
29.3.12	DMM Destination x Region 1 (DMMDESTxREG1) .....	1844
29.3.13	DMM Destination x Blocksize 1 (DMMDESTxBL1) .....	1845
29.3.14	DMM Destination x Region 2 (DMMDESTxREG2) .....	1846
29.3.15	DMM Destination x Blocksize 2 (DMMDESTxBL2) .....	1847
29.3.16	DMM Pin Control 0 (DMMPC0) .....	1848
29.3.17	DMM Pin Control 1 (DMMPC1) .....	1850
29.3.18	DMM Pin Control 2 (DMMPC2) .....	1852
29.3.19	DMM Pin Control 3 (DMMPC3) .....	1853
29.3.20	DMM Pin Control 4 (DMMPC4) .....	1855
29.3.21	DMM Pin Control 5 (DMMPC5) .....	1857
29.3.22	DMM Pin Control 6 (DMMPC6) .....	1859
29.3.23	DMM Pin Control 7 (DMMPC7) .....	1861
29.3.24	DMM Pin Control 8 (DMMPC8) .....	1863
<b>30</b>	<b>RAM Trace Port (RTP) .....</b>	<b>1865</b>
30.1	Overview .....	1866
30.1.1	Features .....	1866



30.1.2	Block Diagram .....	1866
30.2	Module Operation .....	1867
30.2.1	Trace Mode .....	1867
30.2.2	Direct Data Mode (DDM) .....	1868
30.2.3	Trace Regions .....	1869
30.2.4	Overflow/Empty Handling .....	1871
30.2.5	Signal Description .....	1871
30.2.6	Data Rate .....	1872
30.3	GIO Function .....	1873
30.4	Control Registers .....	1874
30.4.1	RTP Global Control Register (RTPGLBCTRL) .....	1875
30.4.2	RTP Trace Enable Register (RTPTRENA) .....	1878
30.4.3	RTP Global Status Register (RTPGSR) .....	1879
30.4.4	RTP RAM 1 Trace Region [1:2] Register (RTPRAM1REG[1:2]) .....	1881
30.4.5	RTP RAM 2 Trace Region [1:2] Register (RTPRAM2REG[1:2]) .....	1882
30.4.6	RTP Peripheral Trace Region [1:2] Registers (RTPPERREG[1:2]) .....	1883
30.4.7	RTP Direct Data Mode Write Register (RTPDDMW) .....	1884
30.4.8	RTP Pin Control 0 Register (RTPPC0) .....	1885
30.4.9	RTP Pin Control 1 Register (RTPPC1) .....	1886
30.4.10	RTP Pin Control 2 Register (RTPPC2) .....	1887
30.4.11	RTP Pin Control 3 Register (RTPPC3) .....	1888
30.4.12	RTP Pin Control 4 Register (RTPPC4) .....	1889
30.4.13	RTP Pin Control 5 Register (RTPPC5) .....	1890
30.4.14	RTP Pin Control 6 Register (RTPPC6) .....	1891
30.4.15	RTP Pin Control 7 Register (RTPPC7) .....	1893
30.4.16	RTP Pin Control 8 Register (RTPPC8) .....	1894

## List of Figures

1-1.	Block Diagram .....	88
1-2.	Example: SPIDELAY – 0xFFFF7F448 .....	89
2-1.	Architectural Block Diagram .....	92
2-2.	Clock Test Register (CLKTEST) [offset = FFFF FFF8Ch].....	100
2-3.	LPO and Clock Detection, Untrimmed HF LPO.....	101
2-4.	Memory Map .....	103
2-5.	Hardware Memory Initialization Protocol .....	113
2-6.	SYS Pin Control Register 1 (SYSPC1) [offset = 00] .....	120
2-7.	SYS Pin Control Register 2 (SYSPC2) [offset = 04h].....	121
2-8.	SYS Pin Control Register 3 (SYSPC3) [offset = 08h].....	122
2-9.	SYS Pin Control Register 4 (SYSPC4) [offset = 0Ch] .....	123
2-10.	SYS Pin Control Register 5 (SYSPC5) [offset = 10h].....	124
2-11.	SYS Pin Control Register 6 (SYSPC6) [offset = 14h].....	125
2-12.	SYS Pin Control Register 7 (SYSPC7) [offset = 18h].....	126
2-13.	SYS Pin Control Register 8 (SYSPC8) [offset = 1Ch] .....	127
2-14.	SYS Pin Control Register 9 (SYSPC9) [offset = 20h].....	128
2-15.	Clock Source Disable Register (CSDIS) [offset = 30h] .....	129
2-16.	Clock Source Disable Set Register (CSDISSET) [offset = 34h] .....	130
2-17.	Clock Source Disable Clear Register (CSDISCLR) [offset = 38h].....	131
2-18.	Clock Domain Disable Register (CDDIS) [offset = 3Ch].....	132
2-19.	Clock Domain Disable Set Register (CDDISSET) [offset = 40h] .....	133
2-20.	Clock Domain Disable Clear Register (CDDISCLR) [offset = 44h].....	135
2-21.	GCLK, HCLK, VCLK, and VCLK2 Source Register (GHVSR) [offset = 48h] .....	137
2-22.	Peripheral Asynchronous Clock Source Register (VCLKASRC) [offset = 4Ch] .....	139
2-23.	RTI Clock Source Register (RCLKSRC) [offset = 50h] .....	140
2-24.	Clock Source Valid Status Register (CSVSTAT) [offset = 54h].....	141
2-25.	Memory Self-Test Global Control Register (MSTGCR) [offset = 58h] .....	142
2-26.	Memory Hardware Initialization Global Control Register (MINITGCR) [offset = 5Ch] .....	143
2-27.	MBIST Controller/Memory Initialization Enable Register (MSINENA) [offset = 60h].....	144
2-28.	Memory Self-Test Fail Status Register (MSTFAIL) [offset = 64h] .....	145
2-29.	MSTC Global Status Register (MSTCGSTAT) [offset = 68h] .....	146
2-30.	Memory Hardware Initialization Status Register (MINISTAT) [offset = 6Ch].....	147
2-31.	PLL Control Register 1 (PLLCTL1) [offset = 70h] .....	148
2-32.	PLL Control Register 2 (PLLCTL2) [offset = 74h] .....	149
2-33.	SYS Pin Control Register 10 (SYSPC10) [offset = 78h].....	151
2-34.	Die Identification Register, Lower Word (DIEIDL) [offset = 7Ch] .....	152
2-35.	Die Identification Register, Upper Word (DIEIDH) [offset = 80h] .....	153
2-36.	LPO/Clock Monitor Control Register (LPOMONCTL) [offset = 088h].....	154
2-37.	Clock Test Register (CLKTEST) [offset = 8Ch].....	157
2-38.	DFT Control Register (DFTCTRLREG) [offset = 90h] .....	159
2-39.	DFT Control Register 2 (DFTCTRLREG2) [offset = 94h] .....	160
2-40.	General Purpose Register (GPREG1) [offset = A0h] .....	161
2-41.	Imprecise Fault Status Register (IMPFASTS) [offset = A8h].....	162
2-42.	Imprecise Fault Write Address Register (IMPFTADD) [offset = ACh].....	164
2-43.	System Software Interrupt Request 1 Register (SSIR1) [offset = B0h].....	165
2-44.	System Software Interrupt Request 2 Register (SSIR2) [offset = B4h].....	166
2-45.	System Software Interrupt Request 3 Register (SSIR3) [offset = B8h].....	167

2-46.	System Software Interrupt Request 4 Register (SSIR4) [offset = BCh] .....	168
2-47.	RAM Control Register (RAMGCR) [offset = C0h] .....	169
2-48.	Bus Matrix Module Control Register 1 (BMMCR) [offset = C4h] .....	170
2-49.	CPU Reset Control Register (CPURSTCR) [offset = CCh] .....	171
2-50.	Clock Control Register (CLKCNTRL) [offset = D0h] .....	172
2-51.	ECP Control Register (ECPCNTL) [offset = D4h] .....	173
2-52.	DEV Parity Control Register 1 (DEVCR1) [offset = DCh] .....	174
2-53.	System Exception Control Register (SYSECR) [offset = E0h] .....	175
2-54.	System Exception Status Register (SYSESR) [offset = E4h] .....	176
2-55.	System Test Abort Status Register (SYSTASR) [offset = E8h] .....	177
2-56.	Global Status Register (GLBSTAT) [offset = ECh] .....	178
2-57.	Device Identification Register (DEVID) [offset = F0h] .....	179
2-58.	Software Interrupt Vector Register (SSIVVEC) [offset = F4h] .....	180
2-59.	System Software Interrupt Flag Register (SSIF) [offset = F8h] .....	181
2-60.	PLL Control Register 3 (PLLCTL3) [offset = 00] .....	183
2-61.	CPU Logic BIST Clock Prescaler (STCLKDIV) [offset = 08h] .....	184
2-62.	Clock 2 Control Register (CLK2CNTRL) [offset = 3Ch] .....	185
2-63.	Peripheral Asynchronous Clock Configuration 1 Register (VCLKACON1) [offset = 40h] .....	186
2-64.	Clock Slip Register (CLKSLIP) [offset = 70h] .....	188
2-65.	EFUSE Controller Control Register (EFC_CTLREG) [offset = ECh] .....	189
2-66.	Die Identification Register, Lower Word (DIEIDL_REG0) [offset = F0h] .....	190
2-67.	Die Identification Register, Upper Word (DIEIDH_REG1) [offset = F4h] .....	191
2-68.	Die Identification Register, Lower Word (DIEIDL_REG2) [offset = F8h] .....	192
2-69.	Die Identification Register, Upper Word (DIEIDH_REG3) [offset = FCh] .....	193
2-70.	Peripheral Memory Protection Set Register 0 (PMPROTSET0) [offset = 00] .....	195
2-71.	Peripheral Memory Protection Set Register 1 (PMPROTSET1) [offset = 04h] .....	196
2-72.	Peripheral Memory Protection Clear Register 0 (PMPROTCLR0) [offset = 10h] .....	197
2-73.	Peripheral Memory Protection Clear Register 1 (PMPROTCLR1) [offset = 14h] .....	198
2-74.	Peripheral Protection Set Register 0 (PPROTSET0) [offset = 20h] .....	199
2-75.	Peripheral Protection Set Register 1 (PPROTSET1) [offset = 24h] .....	200
2-76.	Peripheral Protection Set Register 2 (PPROTSET2) [offset = 28h] .....	201
2-77.	Peripheral Protection Set Register 3 (PPROTSET3) [offset = 2Ch] .....	202
2-78.	Peripheral Protection Clear Register 0 (PPROTCLR0) [offset = 40h] .....	203
2-79.	Peripheral Protection Clear Register 1 (PPROTCLR1) [offset = 44h] .....	204
2-80.	Peripheral Protection Clear Register 2 (PPROTCLR2) [offset = 48h] .....	205
2-81.	Peripheral Protection Clear Register 3 (PPROTCLR3) [offset = 4Ch] .....	206
2-82.	Peripheral Memory Power-Down Set Register 0 (PCSPWRDWNSET0) [offset = 60h] .....	207
2-83.	Peripheral Memory Power-Down Set Register 1 (PCSPWRDWNSET1) [offset = 64h] .....	208
2-84.	Peripheral Memory Power-Down Clear Register 0 (PCSPWRDWNCLR0) [offset = 70h] .....	209
2-85.	Peripheral Memory Power-Down Clear Register 1 (PCSPWRDWNCLR1) [offset = 74h] .....	210
2-86.	Peripheral Power-Down Set Register 0 (PSPWRDWNSET0) [offset = 80h] .....	211
2-87.	Peripheral Power-Down Set Register 1 (PSPWRDWNSET1) [offset = 84h] .....	212
2-88.	Peripheral Power-Down Set Register 2 (PSPWRDWNSET2) [offset = 88h] .....	213
2-89.	Peripheral Power-Down Set Register 3 (PSPWRDWNSET3) [offset = 8Ch] .....	214
2-90.	Peripheral Power-Down Clear Register 0 (PSPWRDWNCLR0) [offset = A0h] .....	215
2-91.	Peripheral Power-Down Clear Register 1 (PSPWRDWNCLR1) [offset = A4h] .....	216
2-92.	Peripheral Power-Down Clear Register 2 (PSPWRDWNCLR2) [offset = A8h] .....	217
2-93.	Peripheral Power-Down Clear Register 3 (PSPWRDWNCLR) [offset = ACh] .....	218
3-1.	PMM Block Diagram .....	220

3-2.	Core Power Domains.....	222
3-3.	Logic Power Domain Control Register (LOGICPDPWRCTRL0) [offset = 00] .....	227
3-4.	Memory Power Domain Control Register 0 (MEMDPDPWRCTRL0) [offset = 10h] .....	229
3-5.	Power Domain Clock Disable Register (PDCLKDISREG) [offset = 20h] .....	230
3-6.	Power Domain Clock Disable Set Register (PDCLKDISSETREG) [offset = 24h] .....	231
3-7.	Power Domain Clock Disable Clear Register (PDCLKDISCLRREG) [offset = 28h] .....	232
3-8.	Logic Power Domain PD2 Power Status Register (LOGICPDPWRSTAT0) [offset = 40h].....	233
3-9.	Logic Power Domain PD3 Power Status Register (LOGICPDPWRSTAT1) [offset = 44h].....	234
3-10.	Logic Power Domain PD4 Power Status Register (LOGICPDPWRSTAT2) [offset = 48h].....	235
3-11.	Logic Power Domain PD5 Power Status Register (LOGICPDPWRSTAT3) [offset = 4Ch] .....	236
3-12.	Memory Power Domain RAM_PD1 Power Status Register (MEMDPDPWRSTAT0) [offset = 80h].....	237
3-13.	Memory Power Domain RAM_PD2 Power Status Register (MEMDPDPWRSTAT1) [offset = 84h].....	238
3-14.	Memory Power Domain RAM_PD3 Power Status Register (MEMDPDPWRSTAT2) [offset = 88h].....	239
3-15.	Global Control Register 1 (GLOBALCTRL1) [offset = A0h] .....	240
3-16.	Global Status Register (GLOBALSTAT) [offset = A8h] .....	241
3-17.	PSCON Diagnostic Compare Key Register (PRCKEYREG) [offset = ACh] .....	242
3-18.	LogicPD PSCON Diagnostic Compare Status Register 1 (LPDDCSTAT1) [offset = B0h].....	243
3-19.	LogicPD PSCON Diagnostic Compare Status Register 2 (LPDDCSTAT2) [offset = B4h].....	244
3-20.	Memory PD PSCON Diagnostic Compare Status Register 1 (MPDDCSTAT1) [offset = B8h] .....	245
3-21.	Memory PD PSCON Diagnostic Compare Status Register 2 (MPDDCSTAT2) [offset = BCh] .....	246
3-22.	Isolation Diagnostic Status Register (ISODIAGSTAT) [offset = C0h] .....	247
4-1.	PINMMR10 Control Register, Address = FFFF EB38h .....	251
4-2.	Output Multiplexing Example .....	252
4-3.	Input Multiplexing Example .....	253
4-4.	REVISION_REG: Revision Register (Offset = 00) .....	256
4-5.	BOOT_REG: Boot Mode Register (Offset = 20h) .....	257
4-6.	KICK_REG0: Kicker Register 0 (Offset = 38h) .....	258
4-7.	KICK_REG1: Kicker Register 1 (Offset = 3Ch) .....	259
4-8.	ERR_RAW_STATUS_REG: Error Raw Status / Set Register (Offset = E0h) .....	260
4-9.	ERR_ENABLED_STATUS_REG: Error Enabled Status / Clear Register (Offset = E4h) .....	261
4-10.	ERR_ENABLE_REG: Error Signaling Enable Register (Offset = E8h) .....	262
4-11.	ERR_ENABLE_CLR_REG: Error Signaling Enable Clear Register (Offset = ECh) .....	263
4-12.	FAULT_ADDRESS_REG: Fault Address Register (Offset = F4h) .....	264
4-13.	FAULT_STATUS_REG: Fault Status Register (Offset = F8h) .....	265
4-14.	FAULT_CLEAR_REG: Fault Clear Register (Offset = FCh) .....	266
4-15.	PINMMRnn: Pin Multiplexing Control Registers (Offset = 110h - 188h).....	267
5-1.	Flash Option Control Register (FRDCNTL) [offset = FFF8 7000h] .....	278
5-2.	32-bit, 4 RowsFlash Error Detection and Correction Control Register 1 (FEDACCTRL1) [offset = FFF8 7008h] .....	279
5-3.	Flash Error Correction and Correction Control Register 2 (FEDACCTRL2) [offset = FFF8 700Ch].....	281
5-4.	Flash Un-correctable Error Address (FUNC_ERR_ADD) [offset = FFF8 7020h] .....	282
5-5.	Flash Bank Protection Register (FBPROT) [offset = FFF8 7030h].....	283
5-6.	Flash Bank Sector Enable Register (FBSE) [offset = FFF8 7034h].....	284
5-7.	Flash Bank Access Control Register (FBAC) [offset = FFF8 703Ch].....	285
5-8.	Flash Bank Fallback Power Register (FBFALLBACK) [offset = FFF8 7040h] .....	286
5-9.	Flash Pump Access Control Register 1 (FPAC1) [offset = FFF8 7048h].....	287
5-10.	Flash Pump Access Control Register 2 (FPAC2) [offset = FFF8 704Ch] .....	288
5-11.	Flash Module Access Control Register (FMAC) [offset = FFF8 7050h] .....	289
5-12.	Flash Module Status Register (FMSTAT) [offset = FFF8 7054h] .....	290

5-13.	FSM Register Write Enable (FSM_WR_ENA) [offset = FFF8 7288h] .....	292
5-14.	EEPROM Emulation Configuration Register (EEPROM_CONFIG) [offset = FFF8 72B8h] .....	293
6-1.	TCRAM Module Connections .....	296
6-2.	RAM Memory Map .....	298
6-3.	TCRAM Module Control Register (RAMCTRL) [offset = 00] .....	303
6-4.	TCRAM Module Single-Bit Error Correction Threshold Register (RAMTHRESHOLD) [offset = 04h] .....	305
6-5.	TCRAM Module Single-Bit Error Occurrences Counter Register (RAMOCCUR) [offset = 08h].....	306
6-6.	TCRAM Module Interrupt Control Register (RAMINTCTRL) [offset = 0Ch] .....	307
6-7.	TCRAM Module Error Status Register (RAMERRSTATUS) [offset = 10h].....	308
6-8.	TCRAM Module Single-Bit Error Address Register (RAMSERRADDR) [offset = 14h] .....	309
6-9.	TCRAM Module Uncorrectable Error Address Register (RAMUERRADDR) [offset = 1Ch] .....	310
6-10.	TCRAM Module Test Mode Control Register (RAMTEST) [offset = 30h] .....	311
6-11.	TCRAM Module Test Mode Vector Register (RAMADDRDECVECT) [offset = 38h] .....	312
6-12.	TCRAM Module Parity Error Address Register (RAMPERRADDR) [offset = 3Ch] .....	313
7-1.	PBIST Block Diagram .....	316
7-2.	PBIST Memory Self-Test Flow Diagram .....	321
7-3.	RAM Configuration Register (RAMT) [offset = 0160h] .....	327
7-4.	Datalogger Register (DLR) [offset = 0164h] .....	328
7-5.	PBIST Activate/ROM Clock Enable Register (PACT) [offset = 0180h] .....	329
7-6.	PBIST ID Register [offset = 184h] .....	330
7-7.	Override Register (OVER) [offset = 0188h] .....	331
7-8.	Fail Status Fail Register 0 (FSRF0) [offset = 0190h].....	332
7-9.	Fail Status Fail Register 1 (FSRF1) [offset = 0194h].....	332
7-10.	Fail Status Count 0 Register (FSRC0) [offset = 0198h] .....	333
7-11.	Fail Status Count Register 1 (FSRC1) [offset = 019Ch] .....	333
7-12.	Fail Status Address 0 Register (FSRA0) [offset = 01A0h].....	334
7-13.	Fail Status Address 1 Register (FSRA1) [offset = 01A4h].....	334
7-14.	Fail Status Data Register 0 (FSRDL0) [offset = 01A8h] .....	335
7-15.	Fail Status Data Register 1 (FSRDL1) [offset = 01B0h] .....	335
7-16.	ROM Mask Register (ROM) [offset = 01C0h] .....	336
7-17.	ROM Algorithm Mask Register (ALGO) [offset = 01C4h].....	337
7-18.	RAM Info Mask Lower Register (RINFOL) [offset = 01C8h] .....	338
7-19.	RAM Info Mask Upper Register (RINFOU) [offset = 01CCh] .....	339
8-1.	STC Block Diagram .....	345
8-2.	Application Self-Test Flow Chart .....	347
8-3.	STC Global Control Register0 (STCGCR0) .....	350
8-4.	STCGCR1 (STC Global Control Register1) .....	351
8-5.	Self-Test Run Timeout Counter Preload Register (STCTPR).....	352
8-6.	STC Current ROM Address Register (STC_CADDR).....	353
8-7.	STC Current Interval Count Register (STCCICR) .....	354
8-8.	Self-Test Global Status Register (STCGSTAT) .....	355
8-9.	Self-Test Fail Status Register (STCFSTAT) .....	356
8-10.	CPU1 Current MISR Register (CPU1_CURMISR3).....	357
8-11.	CPU1 Current MISR Register (CPU1_CURMISR2).....	357
8-12.	CPU1 Current MISR Register (CPU1_CURMISR1).....	357
8-13.	CPU1 Current MISR Register (CPU1_CURMISR0).....	357
8-14.	CPU2 Current MISR Register (CPU2_CURMISR3).....	359
8-15.	CPU2 Current MISR Register (CPU2_CURMISR2).....	359
8-16.	CPU2 Current MISR Register (CPU2_CURMISR1).....	359

8-17.	CPU2 Current MISR Register (CPU2_CURMISR0.....)	359
9-1.	Block Diagram .....	364
9-2.	CCM-R4F Status Register (CCMSR) (Offset = FFFF F600h) .....	369
9-3.	CCM-R4F Key Register (CCMKEYR) (Offset = FFFF F604h) .....	370
10-1.	Clock Path From Oscillator Through PLL To Device .....	373
10-2.	Clock Generation Path .....	374
10-3.	Oscillator Implementation .....	375
10-4.	Operation of the FM-PLL module .....	379
10-5.	PLL Slip Detection & Reset/Bypass Block Diagram .....	384
10-6.	PLL Control 1 Register (PLLCTL1) [offset = FFFF FF70h].....	387
10-7.	PLL Control 2 Register (PLLCTL2) [offset = FFFF FF74h].....	389
10-8.	PLL Control 1 Register (PLLCTL3) [offset = FFFF E100h] .....	391
10-9.	PLL Clock Slip Control Register (CLKSLIP) [offset = FFFF E170h] .....	393
10-10.	General Purpose Register1 (GPREG1) [offset = FFFF FFA0h] .....	394
10-11.	SSW PLL BIST Control Register 1 (SSWPLL1) [offset = 24h] .....	395
10-12.	SSW PLL BIST Control Register 2 (SSWPLL2) [offset = 28h].....	397
10-13.	SSW PLL BIST Control Register 3 (SSWPLL3) [offset = 2Ch] .....	398
10-14.	LPO/Clock Monitor Control Register (LPOMONCTL) [offset = 88h] .....	399
10-15.	Clock Test Register (CLKTEST) [offset = 8Ch].....	402
10-16.	Basic PLL Circuit .....	403
10-17.	PFD Timing .....	403
10-18.	PLL Modulation Block Diagram .....	404
10-19.	Frequency vs. Time .....	405
11-1.	DCC Block Diagram .....	410
11-2.	Counter Relationship .....	411
11-3.	Clock1 slower than Clock0 results in an error and stops counting .....	412
11-4.	Clock1 faster than Clock0 results in an error and stops counting .....	412
11-5.	Clock1 not present results in an error and stops counting .....	413
11-6.	Clock0 not present results in an error and stops counting .....	413
11-7.	DCC Control Register (DCCGCTRL) [offset = 00] .....	416
11-8.	DCC Revision Id Register (DCCREV) [offset = 4h] .....	417
11-9.	DCC Counter0 Seed Register (DCCCNT0SEED) [offset = 8h] .....	418
11-10.	DCC Valid0 Seed Register (DCCVALID0SEED) [offset = Ch] .....	419
11-11.	DCC Counter1 Seed Register (DCCCNT1SEED) [offset = 10h] .....	420
11-12.	DCC Status Register (DCCSTAT) [offset = 14h] .....	421
11-13.	DCC Counter0 Value Register (DCCCNT0) [offset = 18h] .....	422
11-14.	DCC Valid0 Value Register (DCCVALID0) [offset = 1Ch] .....	423
11-15.	DCC Counter1 Value Register (DCCCNT1) [offset = 20h] .....	424
11-16.	DCC Counter1 Clock Source Selection Register (DCCCNT1CLKSRC) [offset = 24h] .....	425
11-17.	DCC Counter1 Clock Source Selection Register (DCCCNT1CLKSRC) [offset = 24h] .....	426
12-1.	Block Diagram .....	428
12-2.	Interrupt Response Handling .....	429
12-3.	<b>ERROR</b> Pin Response Handling .....	429
12-4.	<b>ERROR</b> Pin Timing - Example 1 .....	431
12-5.	<b>ERROR</b> Pin Timing - Example 2 .....	431
12-6.	<b>ERROR</b> Pin Timing - Example 3 .....	431
12-7.	<b>ERROR</b> Pin Timing - Example 4 .....	431
12-8.	<b>ERROR</b> Pin Timing - Example 5 .....	431
12-9.	<b>ERROR</b> Pin Timing - Example 6 .....	432



12-10. ESM Initialization .....	433
12-11. ESM Enable <b>ERROR</b> Pin Action/Response Register 1 (ESMEEPAPR1) .....	435
12-12. ESM Disable <b>ERROR</b> Pin Action/Response Register 1 (ESMDEPAPR1) .....	436
12-13. ESM Interrupt Enable Set/Status Register 1 (ESMIESR1) .....	437
12-14. ESM Interrupt Enable Clear/Status Register 1 (ESMIECR1) .....	438
12-15. ESM Interrupt Level Set/Status Register 1 (ESMILSR1) .....	439
12-16. ESM Interrupt Level Clear/Status Register 1 (ESMILCR1).....	440
12-17. ESM Status Register 1 (ESMSR1) .....	441
12-18. ESM Status Register 2 (ESMSR2) .....	442
12-19. ESM Status Register 3 (ESMSR3) .....	443
12-20. ESM <b>ERROR</b> Pin Status Register (ESMEPSR) .....	444
12-21. ESM Interrupt Offset High Register (ESMIOFFHR) .....	445
12-22. ESM Interrupt Offset Low Register (ESMIOFFLR) .....	446
12-23. ESM Low-Time Counter Register (ESMLTCR) .....	447
12-24. ESM Low-Time Counter Preload Register (ESMLTCPR) .....	448
12-25. ESM Error Key Register (ESMEKR) .....	449
12-26. ESM Status Shadow Register 2 .....	450
12-27. ESM Influence <b>ERROR</b> Pin Set/Status Register 4 (ESMIEPSR4) .....	451
12-28. ESM Influence <b>ERROR</b> Pin Clear/Status Register 4 (ESMIEPSR4).....	452
12-29. ESM Interrupt Enable Set/Status Register 4 (ESMIESR4) .....	453
12-30. ESM Interrupt Enable Clear/Status Register 4 (ESMIECR4) .....	454
12-31. ESM Interrupt Level Set/Status Register 4 (ESMILSR4) .....	455
12-32. ESM Interrupt Level Clear/Status Register 4 (ESMILCR4).....	456
12-33. ESM Status Register 4 (ESMSR4) .....	457
13-1. RTI Block Diagram .....	461
13-2. Counter Block Diagram .....	462
13-3. Compare Unit Block Diagram (shows only 1 of 4 blocks for simplification) .....	464
13-4. Timebase Control .....	465
13-5. Clock Detection Scheme .....	465
13-6. Switch to NTUx .....	466
13-7. Missing NTUx Signal Example .....	467
13-8. Digital Watchdog .....	467
13-9. DWD Operation .....	468
13-10. Digital Windowed Watchdog Timing Example .....	469
13-11. Digital Windowed Watchdog Operation Example (25% Window) .....	469
13-12. RTI Global Control Register (RTIGCTRL) [offset = 00] .....	473
13-13. RTI Timebase Control Register (RTITBCTRL) [offset = 04h] .....	474
13-14. RTI Capture Control Register (RTICAPCTRL) [offset = 08h] .....	475
13-15. RTI Compare Control Register (RTICOMPCTRL) [offset = 0Ch].....	476
13-16. RTI Free Running Counter 0 Register (RTIFRC0) [offset = 10h].....	477
13-17. RTI Up Counter 0 Register (RTIUC0) [offset = 14h] .....	478
13-18. RTI Compare Up Counter 0 Register (RTICPUC0) [offset = 18h].....	479
13-19. RTI Capture Free Running Counter 0 Register (RTICAFRC0) [offset = 20h].....	480
13-20. RTI Capture Up Counter 0 Register (RTICAUC0) [offset = 24].....	481
13-21. RTI Free Running Counter 1 Register (RTIFRC1) [offset = 30h].....	482
13-22. RTI Up Counter 1 Register (RTIUC1) [offset = 34h] .....	483
13-23. RTI Compare Up Counter 1 Register (RTICPUC1) [offset = 38h].....	484
13-24. RTI Capture Free Running Counter 1 Register (RTICAFRC1) [offset = 40h].....	485
13-25. RTI Capture Up Counter 1 Register (RTICAUC1) [offset = 44h] .....	486

13-26. RTI Compare 0 Register (RTICOMP0) [offset = 50h] .....	487
13-27. RTI Update Compare 0 Register (RTIUDCP0) [offset = 54h] .....	488
13-28. RTI Compare 1 Register (RTICOMP1) [offset = 58h] .....	489
13-29. RTI Update Compare 1 Register (RTIUDCP1) [offset = 5Ch] .....	490
13-30. RTI Compare 2 Register (RTICOMP2) [offset = 60h] .....	491
13-31. RTI Update Compare 2 Register (RTIUDCP2) [offset = 64h] .....	492
13-32. RTI Compare 3 Register (RTICOMP3) [offset = 68h] .....	493
13-33. RTI Update Compare 3 Register (RTIUDCP3) [offset = 6Ch] .....	494
13-34. RTI Timebase Low Compare Register (RTITBLCOMP) [offset = 70h] .....	495
13-35. RTI Timebase High Compare Register (RTITBHCOMP) [offset = 74h] .....	496
13-36. RTI Set Interrupt Control Register (RTISETINTENA) [offset = 80h] .....	497
13-37. RTI Clear Interrupt Control Register (RTICLEARINTENA) [offset = 84h] .....	499
13-38. RTI Interrupt Flag Register (RTIINTFLAG) [offset = 88h] .....	501
13-39. Digital Watchdog Control Register (RTIDWDCTRL) [offset = 90h] .....	503
13-40. Digital Watchdog Preload Register (RTIDWDPRLD) [offset = 94h] .....	504
13-41. Watchdog Status Register (RTIWDSTATUS) [offset = 98h] .....	505
13-42. RTI Watchdog Key Register (RTIDWDKEY) [offset = 9Ch] .....	506
13-43. RTI Watchdog Down Counter Register (RTIDWDCNTR) [offset = A0h] .....	507
13-44. Digital Windowed Watchdog Reaction Control (RTIWWDRXNCTRL) [offset = A4h] .....	508
13-45. Digital Windowed Watchdog Window Size Control (RTIWWDSIZECTRL) [offset = A8h] .....	509
14-1. CRC Controller Block Diagram For One Channel .....	513
14-2. MCRC Controller Block Diagram For One Channel .....	514
14-3. LFSR .....	516
14-4. AUTO Mode Using Hardware Timer Trigger .....	519
14-5. AUTO Mode With Software CPU Trigger .....	519
14-6. Semi-CPU Mode With Hardware Timer Trigger .....	520
14-7. Timeout Example 1 .....	522
14-8. Timeout Example 2 .....	523
14-9. Timeout Example 3 .....	523
14-10. CRC Global Control Register 0 (CRC_CTRL0) [offset = 00] .....	531
14-11. CRC Global Control Register 1 (CRC_CTRL1) [offset = 08h] .....	532
14-12. CRC Global Control Register 2 (CRC_CTRL2) [offset = 10h] .....	533
14-13. CRC Interrupt Enable Set Register (CRC_INTS) [offset = 18h] .....	535
14-14. CRC Interrupt Enable Reset Register (CRC_INTR) [offset = 20h] .....	540
14-15. CRC Interrupt Status Register (CRC_STATUS) [offset = 28h] .....	545
14-16. CRC Interrupt Offset (CRC_INT_OFFSET_REG) [offset = 30h] .....	548
14-17. CRC Busy Register (CRC_BUSY) [offset = 38h] .....	549
14-18. CRC Pattern Counter Preload Register 1 (CRC_PCOUNT_REG1) [offset = 40h] .....	550
14-19. CRC Sector Counter Preload Register 1 (CRC_SCOUNT_REG1) [offset = 44h] .....	551
14-20. CRC Current Sector Preload Register 1 (CRC_CURSEC_REG1) [offset = 48h] .....	552
14-21. CRC Channel 1 Watchdog Timeout Preload Register A (CRC_WDTOPLD1) [offset = 4Ch] .....	553
14-22. CRC Channel 1 Block Complete Timeout Preload Register B (CRC_BCTOPLD1) [offset = 50h] .....	554
14-23. Channel 1 PSA Signature Low Register (PSA_SIGREGL1) [offset = 60h] .....	555
14-24. Channel 1 PSA Signature High Register (PSA_SIGREGH1) [offset = 64h] .....	556
14-25. Channel 1 CRC Value Low Register (CRC_REGL1) [offset = 68h] .....	557
14-26. Channel 1 CRC Value High Register (CRC_REGH1) [offset = 6Ch] .....	558
14-27. Channel 1 PSA Sector Signature Low Register (PSA_SECSIGREGL1) [offset = 70h] .....	559
14-28. Channel 1 PSA Sector Signature High Register (PSA_SECSIGREGH1) [offset = 74h] .....	560
14-29. Channel 1 Raw Data Low Register (RAW_DATAREGL1) [offset = 78h] .....	561



14-30. Channel 1 Raw Data High Register (RAW_DATAREGH1) [offset = 7Ch] .....	562
14-31. CRC Pattern Counter Preload Register 2 (CRC_PCOUNT_REG2) [offset = 80h].....	563
14-32. CRC Sector Counter Preload Register 2 (CRC_SCOUNT_REG2) [offset = 84h].....	564
14-33. CRC Current Sector Register 2 (CRC_CURSEC_REG2) [offset = 88h] .....	565
14-34. CRC Channel 2 Watchdog Timeout Preload Register A (CRC_WDTPLD2) [offset = 8Ch] .....	566
14-35. CRC Channel 2 Block Complete Timeout Preload Register B (CRC_BCTOPLD2) [offset = 90h].....	567
14-36. Channel 2 PSA Signature Low Register (PSA_SIGREGL2) [offset = A0h] .....	568
14-37. Channel 2 PSA Signature High Register (PSA_SIGREGH2) [offset = A4h] .....	569
14-38. Channel 2 CRC Value Low Register (CRC_REGL2) [offset = A8h] .....	570
14-39. Channel 2 CRC Value High Register (CRC_REGH2) [offset = ACh].....	571
14-40. Channel 2 PSA Sector Signature Low Register (PSA_SECSIGREGL2) [offset = B0h].....	572
14-41. Channel 2 PSA Sector Signature High Register (PSA_SECSIGREGH2) [offset = B4h].....	573
14-42. Channel 2 Raw Data Low Register (RAW_DATAREGL2) [offset = B8h] .....	574
14-43. Channel 2 Raw Data High Register (RAW_DATAREGH2) [offset = BCh] .....	575
14-44. CRC Pattern Counter Preload Register 3 (CRC_PCOUNT_REG3) [offset = C0h] .....	576
14-45. CRC Sector Counter Preload Register 3 (CRC_SCOUNT_REG3) [offset = C4h] .....	577
14-46. CRC Current Sector Register 3 (CRC_CURSEC_REG3) [offset = C8h].....	578
14-47. CRC Channel 3 Watchdog Timeout Preload Register A (CRC_WDTPLD3) [offset = CCh].....	579
14-48. CRC Channel 3 Block Complete Timeout Preload Register B (CRC_BCTOPLD3) [offset = D0h] .....	580
14-49. Channel 3 PSA Signature Low Register (PSA_SIGREGL3) [offset = E0h] .....	581
14-50. Channel 3 PSA Signature High Register (PSA_SIGREGH3) [offset = E4h] .....	582
14-51. Channel 3 CRC Value Low Register (CRC_REGL3) [offset = E8h] .....	583
14-52. Channel 3 CRC Value High Register (CRC_REGH3) [offset = ECh].....	584
14-53. Channel 3 PSA Sector Signature Low Register (PSA_SECSIGREGL3) [offset = F0h].....	585
14-54. Channel 3 PSA Sector Signature High Register (PSA_SECSIGREGH3) [offset = F4h].....	586
14-55. Channel 3 Raw Data Low Register (RAW_DATAREGL3) [offset = F8h].....	587
14-56. Channel 3 Raw Data High Register (RAW_DATAREGH3) [offset = FCh] .....	588
14-57. CRC Pattern Counter Preload Register 4 (CRC_PCOUNT_REG4) [offset = 100h] .....	589
14-58. CRC Sector Counter Preload Register 4 (CRC_SCOUNT_REG4) [offset = 104h] .....	590
14-59. CRC Current Sector Register 4 (CRC_CURSEC_REG4) [offset = 108h].....	591
14-60. CRC Channel 4 Watchdog Timeout Preload Register A (CRC_WDTPLD4) [offset = 10Ch].....	592
14-61. CRC Channel 4 Block Complete Timeout Preload Register B (CRC_BCTOPLD4) [offset = 110h] .....	593
14-62. Channel 4 PSA Signature Low Register (PSA_SIGREGL4) [offset = 120h].....	594
14-63. Channel 4 PSA Signature High Register (PSA_SIGREGH4) [offset = 124h].....	595
14-64. Channel 4 CRC Value Low Register (CRC_REGL4) [offset = 128h] .....	596
14-65. Channel 4 CRC Value High Register (CRC_REGH4) [offset = 12Ch].....	597
14-66. Channel 4 PSA Sector Signature Low Register (PSA_SECSIGREGL4) [offset = 130h].....	598
14-67. Channel 4 PSA Sector Signature High Register (PSA_SECSIGREGH4) [offset = 134h].....	599
14-68. Channel 4 Raw Data Low Register (RAW_DATAREGL4) [offset = 138h] .....	600
14-69. Channel 4 Raw Data High Register (RAW_DATAREGH4) [offset = 13Ch].....	601
14-70. Data Bus Selection Register (MCRC_TRACE_BUS_SEL) [offset = 140h] .....	602
15-1. Device Level Interrupt Block Diagram .....	605
15-2. VIM Interrupt Handling Block Diagram .....	607
15-3. VIM Channel Mapping .....	608
15-4. VIM in Default State .....	609
15-5. VIM in a Programmed State .....	609
15-6. Interrupt Channel Management .....	610
15-7. VIM Interrupt Address Memory Map .....	612
15-8. Parity Bit Mapping .....	614

15-9. Detail of the IRQ Input .....	615
15-10. Capture Event Sources .....	616
15-11. Interrupt Vector Table Parity Flag Register (PARFLG) [Offset = FFFF FDECh] .....	620
15-12. Interrupt Vector Table Parity Control Register (PARCTL) [Offset = FFFF FDF0h] .....	621
15-13. Address Parity Error Register (ADDERR) [Offset = FFFF FDF4h] .....	622
15-14. Fall-Back Address Parity Error Register (FBPARERR) [Offset = FFFF FDF8h] .....	623
15-15. IRQ Index Offset Vector Register (IRQINDEX) [Offset = FFFF FE00h] .....	625
15-16. FIQ Index Offset Vector Register (FIQINDEX) [Offset = FFFF FE04h] .....	626
15-17. FIQ/IRQ Program Control Register 0 (FIRQPR0) [offset = FFFF FE10h].....	627
15-18. FIQ/IRQ Program Control Register 1 (FIRQPR1) [offset = FFFF FE14h].....	627
15-19. FIQ/IRQ Program Control Register 2 (FIRQPR2) [offset = FFFF FE18h].....	627
15-20. Pending Interrupt Read Location Register 0 (INTREQ0) [offset = FFFF FE20h].....	628
15-21. Pending Interrupt Read Location 1 (INTREQ1) Register [offset = FFFF FE24h].....	628
15-22. Pending Interrupt Read Location 2 (INTREQ2) Register [offset = FFFF FE28h].....	628
15-23. Interrupt Enable Set Register 0 (REQENASET0) [offset = FFFF FE30h] .....	629
15-24. Interrupt Enable Set Register 1 (REQENASET1) [offset = FFFF FE34h] .....	629
15-25. Interrupt Enable Set Register 2 (REQENASET2) [offset = FFFF FE38h] .....	629
15-26. Interrupt Enable Clear Register 0 (REQENACLRO) [offset = FFFF FE40h] .....	630
15-27. Interrupt Enable Clear Register 1 (REQENACL1) [offset = FFFF FE44h] .....	630
15-28. Interrupt Enable Clear Register 2 (REQENACL2) [offset = FFFF FE48h] .....	630
15-29. Wake-Up Enable Set Register 0 (WAKEENASET0) [offset = FFFF FE50h].....	631
15-30. Wake-Up Enable Set Register 1 (WAKEENASET1) [offset = FFFF FE54h].....	631
15-31. Wake-Up Enable Set Register 2 (WAKEENASET2) [offset = FFFF FE58h].....	631
15-32. Wake-Up Enable Clear Register 0 (WAKEENACLRO) [offset = FFFF FE60h] .....	632
15-33. Wake-Up Enable Clear Register 1 (WAKEENACL1) [offset = FFFF FE64h] .....	632
15-34. Wake-Up Enable Clear Register 2 (WAKEENACL2) [offset = FFFF FE68h] .....	632
15-35. IRQ Interrupt Vector Register (IRQVECREG) [Offset = FFFF FE70h] .....	633
15-36. IRQ Interrupt Vector Register (FIQVECREG) [Offset = FFFF FE74h] .....	634
15-37. Capture Event Register (CAPEVT) [Offset = FFFF FE78h] .....	635
15-38. Interrupt Control Registers[0:31] (CHANCTRL[0:23]) [Offset = FFFF FE80h - FFFF FEDCh] .....	636
16-1. DMA Block Diagram .....	640
16-2. Example of a DMA Transfer Using Frame Trigger Source .....	641
16-3. Example of a DMA Transfer Using Block Trigger Source .....	641
16-4. DMA Request Mapping and Control Packet Organization .....	642
16-5. Control Packet Organization and Memory Map.....	643
16-6. DMA Transfer Example 1 .....	645
16-7. DMA Indexing Example 1 .....	645
16-8. DMA Indexing Example 2 .....	646
16-9. Fixed Priority Scheme .....	646
16-10. Example of Priority Queues .....	647
16-11. Example Channel Assignments.....	648
16-12. Example of DMA Data Unpacking .....	649
16-13. Example of DMA Data Packing .....	650
16-14. DMA Interrupts .....	652
16-15. Detailed Interrupt Structure (Frame Transfer Complete Path) .....	652
16-16. Example of Channel Chaining.....	655
16-17. Example of Protection Mechanism .....	656
16-18. Global Control Register (GCTRL) [offset = 00] .....	661
16-19. Channel Pending Register (PEND) [offset = 04h] .....	662

16-20. DMA Status Register (DMASTAT) [offset = 0Ch].....	663
16-21. HW Channel Enable Set and Status Register (HWCHENAS) [offset = 14h] .....	664
16-22. HW Channel Enable Reset and Status Register (HWCHENAR) [offset = 1Ch].....	665
16-23. SW Channel Enable Set and Status Register (SWCHENAS) [offset = 24h].....	666
16-24. SW Channel Enable Reset and Status Register (SWCHENAR) [offset = 2Ch] .....	667
16-25. Channel Priority Set Register (CHPRIOS) [offset = 34h] .....	668
16-26. Channel Priority Reset Register (CHPRIOR) [offset = 3Ch] .....	669
16-27. Global Channel Interrupt Enable Set Register (GCHIENAS) [offset = 44h].....	670
16-28. Global Channel Interrupt Enable Reset Register (GCHIENAR) [offset = 4Ch] .....	671
16-29. DMA Request Assignment Register 0 (DREQASI0) [offset = 54h].....	672
16-30. DMA Request Assignment Register 1 (DREQASI1) [offset = 58h].....	673
16-31. DMA Request Assignment Register 2 (DREQASI2) [offset = 5Ch] .....	674
16-32. DMA Request Assignment Register 3 (DREQASI3) [offset = 60h].....	675
16-33. Port Assignment Register 0 (PAR0) [offset = 94h] .....	676
16-34. Port Assignment Register 1 (PAR1) [offset = 98h] .....	678
16-35. FTC Interrupt Mapping Register (FTCMAP) [offset = B4h].....	680
16-36. LFS Interrupt Mapping Register (LFSMAP) [offset = BCh] .....	681
16-37. HBC Interrupt Mapping Register (HBCMAP) [offset = C4h].....	682
16-38. BTC Interrupt Mapping Register (BTCMAP) [offset = CCh].....	683
16-39. BER Interrupt Mapping Register (BERMAP) [offset = D4h] .....	684
16-40. FTC Interrupt Enable Set (FTCINTENAS) [offset = DCh] .....	685
16-41. FTC Interrupt Enable Reset (FTCINTENAR) [offset = E4h].....	686
16-42. LFS Interrupt Enable Set (LFSINTENAS) [offset = ECh] .....	687
16-43. LFS Interrupt Enable Reset (LFSINTENAR) [offset = F4h] .....	688
16-44. HBC Interrupt Enable Set (HBCINTENAS) [offset = FCh].....	689
16-45. HBC Interrupt Enable Reset (HBCINTENAR) [offset = 104h].....	690
16-46. BTC Interrupt Enable Set (BTCINTENAS) [offset = 10Ch] .....	691
16-47. BTC Interrupt Enable Reset (BTCINTENAR) [offset = 114h] .....	692
16-48. Global Interrupt Flag Register (GINTFLAG) [offset = 11Ch] .....	693
16-49. FTC Interrupt Flag Register (FTCFLAG) [offset = 124h] .....	694
16-50. LFS Interrupt Flag Register (LFSFLAG) [offset = 12Ch].....	695
16-51. HBC Interrupt Flag Register (HBCFLAG) [offset = 134h].....	696
16-52. BTC Interrupt Flag Register (BTCFLAG) [offset = 13Ch].....	697
16-53. BER Interrupt Flag Register (BERFLAG) [offset = 144h].....	698
16-54. FTCA Interrupt Channel Offset Register (FTCAOFFSET) [offset = 14Ch] .....	699
16-55. LFSA Interrupt Channel Offset Register (LFSAOFFSET) [offset = 150h] .....	700
16-56. HBCA Interrupt Channel Offset Register (HBCAOFFSET) [offset = 154h].....	701
16-57. BTCA Interrupt Channel Offset Register (BTCAOFFSET) [offset = 158h] .....	702
16-58. BERA Interrupt Channel Offset Register (BERAOFFSET) [offset = 15Ch] .....	703
16-59. FTCB Interrupt Channel Offset Register (FTCBOFFSET) [offset = 160h] .....	704
16-60. LFSB Interrupt Channel Offset Register (LFSBOFFSET) [offset = 164h] .....	705
16-61. HBCB Interrupt Channel Offset Register (HBCBOFFSET) [offset = 168h].....	706
16-62. BTCB Interrupt Channel Offset Register (BTCBOFFSET) [offset = 16Ch].....	707
16-63. BERB Interrupt Channel Offset Register (BERBOFFSET) [offset = 170h].....	708
16-64. Port Control Register (PTCRL) [offset = 178h].....	709
16-65. RAM Test Control (RTCTRL) [offset = 17Ch] .....	710
16-66. Debug Control (DCTRL) [offset = 180h] .....	711
16-67. Watch Point Register (WPR) [offset = 184h] .....	712
16-68. Watch Mask Register (WMR) [offset = 188h] .....	713

16-69. Port A Active Channel Source Address Register (PAACSADDR) [offset = 18Ch] .....	714
16-70. Port A Active Channel Destination Address Register (PAACDADDR) [offset = 190h] .....	715
16-71. Port A Active Channel Transfer Count Register (PAACTC) [offset = 194h].....	716
16-72. Port B Active Channel Source Address Register (PBACDADDR) [offset = 198h] .....	717
16-73. Port B Active Channel Destination Address Register (PBACDADDR) [offset = 19Ch].....	718
16-74. Port B Active Channel Transfer Count Register (PBACTC) [offset = 1A0h] .....	719
16-75. Parity Control Register (DMAPCR) [offset = 1A8h].....	720
16-76. DMA Parity Error Address Register (DMAPAR) [offset = 1ACh] .....	721
16-77. ....	722
16-78. DMA Memory Protection Status Register (DMAMPST) [offset = 1B4h] .....	724
16-79. DMA Memory Protection Region 0 Start Address Register (DMAMPR0S) [offset = 1B8h] .....	725
16-80. DMA Memory Protection Region 0 End Address Register (DMAMPR0E) [offset = 1BCh] .....	726
16-81. DMA Memory Protection Region 1 Start Address Register (DMAMPR1S) [offset = 1C0h].....	727
16-82. DMA Memory Protection Region 1 End Address Register (DMAMPR1E) [offset = 1C4h].....	728
16-83. DMA Memory Protection Region 2 Start Address Register (DMAMPR2S) [offset = 1C8h].....	729
16-84. DMA Memory Protection Region 2 End Address Register (DMAMPR2E) [offset = 1CCh] .....	730
16-85. DMA Memory Protection Region 3 Start Address Register (DMAMPR3S) [offset = 1D0h].....	731
16-86. DMA Memory Protection Region 3 End Address Register (DMAMPR3E) [offset = 1D4h].....	732
16-87. Initial Source Address (ISADDR) [offset = 00] .....	733
16-88. Initial Destination Address Register (IDADDR) [offset = 04h].....	734
16-89. Initial Transfer Count Register (ITCOUNT) [offset = 08h] .....	735
16-90. Channel Control Register (CHCTRL) [offset = 10h] .....	736
16-91. Element Index Offset Register (EIOFF) [offset = 14h] .....	737
16-92. Frame Index Offset Register (FIOFF) [offset = 18h] .....	738
16-93. Current Source Address Register (CSADDR) [offset = 800h] .....	739
16-94. Current Destination Address Register (CDADDR) [offset = 804h] .....	740
16-95. Current Transfer Count Register (CTCOUNT) [offset = 808h].....	741
17-1. EMIF Functional Block Diagram .....	745
17-2. Timing Waveform of SDRAM PRE Command .....	749
17-3. EMIF to 2M × 16 × 4 bank SDRAM Interface .....	749
17-4. EMIF to 512K × 16 × 2 bank SDRAM Interface .....	750
17-5. Timing Waveform for Basic SDRAM Read Operation .....	757
17-6. Timing Waveform for Basic SDRAM Write Operation.....	758
17-7. EMIF Asynchronous Interface .....	760
17-8. EMIF to 8-bit/16-bit Memory Interface.....	761
17-9. Common Asynchronous Interface .....	761
17-10. Timing Waveform of an Asynchronous Read Cycle in Normal Mode.....	765
17-11. Timing Waveform of an Asynchronous Write Cycle in Normal Mode.....	767
17-12. Timing Waveform of an Asynchronous Read Cycle in Select Strobe Mode .....	769
17-13. Timing Waveform of an Asynchronous Write Cycle in Select Strobe Mode .....	771
17-14. Asynchronous Read in Page Mode.....	772
17-15. Module ID Register (MIDR).....	779
17-16. Asynchronous Wait Cycle Configuration Register (AWCCR) .....	779
17-17. SDRAM Configuration Register (SDCR).....	781
17-18. SDRAM Refresh Control Register (SDRCR) .....	783
17-19. Asynchronous <i>n</i> Configuration Register (CE <i>n</i> CFG).....	784
17-20. SDRAM Timing Register (SDTIMR) .....	785
17-21. SDRAM Self Refresh Exit Timing Register (SDSRETR) .....	786
17-22. EMIF Interrupt Raw Register (INTRAW).....	787

17-23. EMIF Interrupt Mask Register (INTMSK) .....	788
17-24. EMIF Interrupt Mask Set Register (INTMSKSET) .....	789
17-25. EMIF Interrupt Mask Clear Register (INTMSKCLR).....	790
17-26. Page Mode Control Register (PMCR).....	791
17-27. Example Configuration Interface.....	793
17-28. SDRAM Timing Register (SDTIMR) .....	794
17-29. SDRAM Self Refresh Exit Timing Register (SDSRETR) .....	795
17-30. SDRAM Refresh Control Register (SDRCR) .....	795
17-31. SDRAM Configuration Register (SDCR).....	796
17-32. LH28F800BJE-PTTL90 to EMIF Read Timing Waveforms .....	797
17-33. LH28F800BJE-PTTL90 to EMIF Write Timing Waveforms.....	798
17-34. Asynchronous <i>m</i> Configuration Register ( <i>m</i> = 1, 2) (CE <sub><i>n</i></sub> CFG ( <i>n</i> = 2, 3)).....	799
18-1. Channel Assignments of Two ADC cores .....	802
18-2. ADC Block Diagram .....	803
18-3. FIFO Implementation .....	808
18-4. Format of Conversion Result Read from FIFO, 12-bit ADC .....	808
18-5. Format of Conversion Result Read from FIFO, 10-bit ADC .....	809
18-6. ADC Memory Mapping .....	809
18-7. Format of Conversion Result Directly Read from ADC RAM, 12-bit ADC .....	810
18-8. Format of Conversion Result Directly Read from ADC RAM, 10-bit ADC .....	810
18-9. Conversion Results Storage .....	811
18-10. Self-Test and Calibration Logic .....	820
18-11. Mid-point Value Calculation .....	823
18-12. Self-Test and Calibration Logic .....	824
18-13. Timing for Self-Test Mode .....	825
18-14. Timing for Sample Capacitor Discharge Mode .....	826
18-15. ADC Memory Map in Parity Test Mode .....	828
18-16. GPIO Functionality of ADxEVT .....	829
18-17. ADC Reset Control Register (ADRSTCR) .....	833
18-18. ADC Operating Mode Control Register (ADOPMODECR) .....	834
18-19. ADC Clock Control Register (ADCLOCKCR) .....	836
18-20. ADC Calibration Mode Control Register (ADCALCR).....	837
18-21. ADC Event Group Operating Mode Control Register (ADEVMODECR) .....	838
18-22. ADC Group1 Operating Mode Control Register (ADG1MODECR) .....	840
18-23. ADC Group2 Operating Mode Control Register (ADG2MODECR) .....	842
18-24. ADC Event Group Trigger Source Select Register (ADEVSRC) .....	844
18-25. ADC Group1 Trigger Source Select Register (ADG1SRC) .....	845
18-26. ADC Group2 Trigger Source Select Register (ADG2SRC) .....	846
18-27. ADC Event Group Interrupt Enable Control Register (ADEVINTENA) .....	847
18-28. ADC Group1 Interrupt Enable Control Register (ADG1INTENA).....	848
18-29. ADC Group2 Interrupt Enable Control Register (ADG2INTENA).....	849
18-30. ADC Event Group Interrupt Flag Register (ADEVINTFLG) .....	850
18-31. ADC Group1 Interrupt Flag Register (ADG1INTFLG).....	851
18-32. ADC Group2 Interrupt Flag Register (ADG2INTFLG).....	852
18-33. ADC Event Group Threshold Interrupt Control Register (ADEVTHRINTCR) .....	853
18-34. ADC Group1 Threshold Interrupt Control Register (ADG1THRINTCR) .....	854
18-35. ADC Group2 Threshold Interrupt Control Register (ADG2THRINTCR) .....	855
18-36. ADC Event Group DMA Control Register (ADEVDMACR).....	856
18-37. ADC Group1 DMA Control Register (ADG1DMACR) .....	857



18-38. ADC Group2 DMA Control Register (ADG2DMACR) .....	858
18-39. ADC Results Memory Configuration Register (ADBNDCCR) .....	859
18-40. ADC Results Memory Size Configuration Register (ADBNDEND) .....	860
18-41. ADC Event Group Sampling Time Configuration Register (ADEVSAAMP).....	861
18-42. ADC Group1 Sampling Time Configuration Register (ADG1SAMP).....	862
18-43. ADC Group2 Sampling Time Configuration Register (ADG2SAMP).....	863
18-44. ADC Event Group Status Register (ADEVSR).....	864
18-45. ADC Group1 Status Register (ADG1SR) .....	865
18-46. ADC Group2 Status Register (ADG2SR) .....	866
18-47. ADC Event Group Channel Select Register (ADEVSEL).....	867
18-48. ADC Group1 Channel Select Register (ADG1SEL) .....	868
18-49. ADC Group2 Channel Select Register (ADG2SEL) .....	869
18-50. ADC Calibration and Error Offset Correction Register (ADCALR) .....	870
18-51. ADC State Machine Status Register (ADSMSTATE) .....	871
18-52. ADC Channel Last Conversion Value Register (ADLASTCONV) .....	872
18-53. 12-bit ADC Event Group Results' FIFO Register (ADEVBUFFER) .....	873
18-54. 10-bit ADC Event Group Results' FIFO Register (ADEVBUFFER) .....	873
18-55. 12-bit ADC Group1 Results FIFO Register (ADG1BUFFER) .....	874
18-56. 10-bit ADC Group1 Results' FIFO Register (ADG1BUFFER).....	874
18-57. 12-bit ADC Group2 Results FIFO Register (ADG2BUFFER) .....	875
18-58. 10-bit ADC Group2 Results' FIFO Register (ADG2BUFFER).....	875
18-59. ADC ADEVT Pin Direction Control Register (ADEVTDIR) .....	877
18-60. ADC ADEVT Pin Output Value Control Register (ADEVTOUT) .....	878
18-61. ADC ADEVT Pin Input Value Register (ADEVTIN).....	879
18-62. ADC ADEVT Pin Set Register (ADEVTSET).....	880
18-63. ADC ADEVT Pin Clear Register (ADEVTCLR) .....	881
18-64. ADC ADEVT Pin Open Drain Enable Register (ADEVTPDR) .....	882
18-65. ADC ADEVT Pin Pull Control Disable Register (ADEVTPDIS).....	883
18-66. ADC ADEVT Pin Pull Control Select Register (ADEVTPSEL).....	884
18-67. ADC Event Group Sample Cap Discharge Control Register (ADEVSAAMPDISEN) .....	885
18-68. ADC Group1 Sample Cap Discharge Control Register (ADG1SAMPDISEN) .....	886
18-69. ADC Group2 Sample Cap Discharge Control Register (ADG2SAMPDISEN) .....	887
18-70. ADC Magnitude Compare Interrupt Control Registers (ADMAGINTxCR).....	888
18-71. ADC Magnitude Compare Mask Register (ADMAGINTxMASK).....	889
18-72. ADC Magnitude Compare Interrupt Enable Set Register (ADMAGINTENASET) .....	890
18-73. ADC Magnitude Compare Interrupt Enable Clear Register (ADMAGINTENACLAR) .....	891
18-74. ADC Magnitude Compare Interrupt Flag Register (ADMAGINTFLG).....	892
18-75. ADC Magnitude Compare Interrupt Offset Register (ADMAGINTOFF) .....	893
18-76. ADC Event Group FIFO Reset Control Register (ADEVFIFORESETCR).....	894
18-77. ADC Group1 FIFO Reset Control Register (ADG1FIFORESETCR).....	895
18-78. ADC Group2 FIFO Reset Control Register (ADG2FIFORESETCR).....	896
18-79. ADC Event Group RAM Write Address Register (ADEVRAMWRADDR) .....	897
18-80. ADC Group1 RAM Write Address Register (ADG1RAMWRADDR) .....	898
18-81. ADC Group2 RAM Write Address Register (ADG2RAMWRADDR) .....	899
18-82. ADC Parity Control Register (ADPARCR) .....	900
18-83. ADC Parity Error Address Register (ADPARADDR) .....	901
18-84. ADC Power-Up Delay Control Register (ADPWRUPDLYCTRL) .....	902
19-1. N2HET Block Diagram .....	907
19-2. Specialized Timer Micromachine .....	908



19-3. Program Flow Timings .....	909
19-4. Use of the Overflow Interrupt Flag (HETEXC2) .....	910
19-5. Multi-Resolution Operation Flow Example .....	911
19-6. Debug Control Configuration .....	912
19-7. Prescaler Configuration .....	915
19-8. I/O Control .....	919
19-9. N2HET Loop Resolution Structure for Each Bit .....	920
19-10. Loop Resolution Instruction Execution Example .....	920
19-11. HR I/O Architecture .....	921
19-12. Example of HR Structure Sharing for N2HET Pins 0/1 .....	922
19-13. XOR-shared HR I/O .....	922
19-14. Symmetrical PWM with XOR-sharing Output .....	923
19-15. AND-shared HR I/O .....	924
19-16. HR0 to HR1 Digital Loopback Logic: LBTYPE[0] = '0' .....	924
19-17. HR0 to HR1 Analog Loop Back Logic: LBTYPE[0] = '1' .....	925
19-18. N2HET Input Edge Detection .....	925
19-19. ECMP Execution Timings .....	927
19-20. High/Low Resolution Modes for ECMP and PWCNT .....	928
19-21. PCNT Instruction Timing (With Capture Edge After HR Counter Overflow) .....	929
19-22. PCNT Instruction Timing (With Capture Edge Before HR Counter Overflow) .....	929
19-23. WCAP Instruction Timing .....	930
19-24. I/O Block Diagram Including Pull Control Logic .....	930
19-25. N2HET Pin Disable Feature Diagram .....	932
19-26. Suppression Filter Counter Operation .....	933
19-27. Interrupt Functionality on Instruction Level .....	934
19-28. Interrupt Flag/Priority Level Architecture .....	935
19-29. Request Line Assignment Example .....	936
19-30. Operation of N2HET Count Instructions .....	937
19-31. SCNT Count Operation .....	938
19-32. ACNT Period Variation Compensations .....	938
19-33. N2HET Timings Associated with the Gap Flag (ACNT Deceleration) <sup>7</sup> .....	939
19-34. N2HET Timings Associated with the Gap Flag (ACNT Acceleration) .....	939
19-35. Global Configuration Register (HETGCR) .....	942
19-36. Prescale Factor Register (HETPFR) .....	944
19-37. N2HET Current Address (HETADDR) .....	945
19-38. Offset Index Priority Level 1 Register (HETOFF1) .....	946
19-39. Offset Index Priority Level 2 Register (HETOFF2) [offset = FFF7 B810h] .....	947
19-40. Interrupt Enable Set Register (HETINTENAS) .....	948
19-41. Interrupt Enable Clear (HETINTENAC) .....	949
19-42. Exception Control Register (HETEXC1) .....	950
19-43. Exception Control Register 2 (HETEXC2) .....	951
19-44. Interrupt Priority Register (HETPRY) .....	952
19-45. Interrupt Flag Register (HETFLG) .....	953
19-46. AND Share Control Register (HETAND) .....	954
19-47. HR Share Control Register (HETHRSH) .....	955
19-48. XOR Share Control Register (HETXOR) .....	956
19-49. Request Enable Set Register (HETREQENS) .....	957
19-50. Request Enable Clear Register (HETREQENC) .....	958
19-51. Request Destination Select Register (HETREQDS) [offset = FFF7 B844h] .....	959

19-52. N2HET Direction Register (HETDIR).....	960
19-53. N2HET Data Input Register (HETDIN).....	961
19-54. N2HET Data Output Register (HETDOOUT).....	962
19-55. N2HET Data Set Register (HETDSET) .....	963
19-56. N2HET Data Clear Register (HETDCLR).....	964
19-57. N2HET Open Drain Register (HETPDR) .....	965
19-58. N2HET Pull Disable Register (HETPULDIS) .....	966
19-59. N2HET Pull Select Register (HETPSL) .....	967
19-60. Parity Control Register (HETPCR).....	968
19-61. Parity Address Register (HETPAR) .....	969
19-62. Parity Pin Register (HETPPR) .....	970
19-63. Suppression Filter Preload Register (HETSFPRLD) .....	971
19-64. Suppression Filter Enable Register (HETSFENA).....	972
19-65. Loop Back Pair Select Register (HETLBPSEL).....	973
19-66. Loop Back Pair Direction Register (HETLBPDIR) .....	974
19-67. N2HET Pin Disable Register (HETPINDIS) .....	975
19-68. ACMP Program Field (P31:P0) .....	981
19-69. ACMP Control Field (C31:C0).....	981
19-70. ACMP Data Field (D31:D0).....	981
19-71. ACNT Program Field (P31:P0) .....	983
19-72. ACNT Control Field (C31:C0) .....	983
19-73. ACNT Data Field (D31:D0) .....	983
19-74. ADCNST Program Field (P31:P0) .....	986
19-75. ADCNST Control Field (C31:C0) .....	986
19-76. ADCNST Data Field (D31:D0) .....	986
19-77. ADCNST Operation If Remote Data Field[31:7] Is Not Zero .....	987
19-78. ADCNST Operation if Remote Data Field [31:7] Is Zero .....	987
19-79. ADC, ADD, AND, OR, SBB, SUB, XOR Program Field (P31:P0) .....	988
19-80. ADC, ADD, AND, OR, SBB, SUB, XOR Control Field (C31:C0) .....	988
19-81. ADC, ADD, AND, OR, SBB, SUB, XOR Data Field (D31:D0) .....	988
19-82. Shift Encoding .....	990
19-83. ADM32 Program Field (P31:P0) .....	995
19-84. ADM32 Control Field (C31:C0).....	995
19-85. ADM32 Data Field (D31:D0).....	995
19-86. ADM32 Add and Move Operation for IM&REGTOREM (Case 00) .....	997
19-87. ADM32 Add and Move Operation for REM&REGTOREG (Case 01) .....	997
19-88. APCNT Program Field (P31:P0) .....	998
19-89. APCNT Control Field (C31:C0) .....	998
19-90. APCNT Data Field (D31:D0) .....	998
19-91. BR Program Field (P31:P0) .....	1001
19-92. BR Control Field (C31:C0) .....	1001
19-93. BR Data Field (D31:D0) .....	1001
19-94. CNT Program Field (P31:P0) .....	1003
19-95. CNT Control Field (C31:C0) .....	1003
19-96. CNT Data Field (D31:D0) .....	1003
19-97. DADM64 Program Field (P31:P0).....	1006
19-98. DADM64 Control Field (C31:C0) .....	1006
19-99. DADM64 Data Field (D31:D0) .....	1007
19-100. DADM64 Add and Move Operation .....	1007

19-101. DJZ Program Field (P31:P0) .....	1008
19-102. DJZ Control Field (C31:C0).....	1008
19-103. DJZ Data Field (D31:D0).....	1008
19-104. ECMP Program Field (P31:P0).....	1010
19-105. ECMP Control Field (C31:C0).....	1010
19-106. ECMP Data Field (D31:D0).....	1010
19-107. ECNT Program Field (P31:P0) .....	1013
19-108. ECNT Control Field (C31:C0) .....	1013
19-109. ECNT Data Field (D31:D0) .....	1013
19-110. MCMP Program Field (P31:P0) .....	1015
19-111. MCMP Control Field (C31:C0) .....	1015
19-112. MCMP Data Field (D31:D0) .....	1015
19-113. MOV32 Program Field (P31:P0) .....	1018
19-114. MOV32 Control Field (C31:C0).....	1018
19-115. MOV32 Data Field (D31:D0).....	1018
19-116. MOV32 Move Operation for IMTOREG (Case 00) .....	1019
19-117. MOV32 Move Operation for IMTOREG&REM (Case 01) .....	1020
19-118. MOV32 Move Operation for REGTOREM (Case 10) .....	1020
19-119. MOV32 Move Operation for REMTOREG (Case 11) .....	1020
19-120. MOV64 Program Field (P31:P0) .....	1022
19-121. MOV64 Control Field (C31:C0).....	1022
19-122. MOV64 Data Field (D31:D0).....	1023
19-123. MOV64 Move Operation .....	1023
19-124. PCNT Program Field (P31:P0) .....	1025
19-125. PCNT Control Field (C31:C0) .....	1025
19-126. PCNT Data Field (D31:D0) .....	1025
19-127. PWCNT Program Field (P31:P0) .....	1028
19-128. PWCNT Control Field (C31:C0).....	1028
19-129. PWCNT Data Field (D31:D0).....	1028
19-130. RADM64 Program Field (P31:P0) .....	1031
19-131. RADM64 Control Field (C31:C0).....	1031
19-132. RADM64 Data Field (D31:D0).....	1032
19-133. RADM64 Add and Move Operation .....	1032
19-134. RCNT Program Field (P31:P0).....	1034
19-135. RCNT Control Field (C31:C0) .....	1034
19-136. RCNT Data Field (D31:D0) .....	1034
19-137. SCMP Program Field (P31:P0).....	1036
19-138. SCMP Control Field (C31:C0).....	1036
19-139. SCMP Data Field (D31:D0).....	1036
19-140. SCNT Program Field (P31:P0) .....	1038
19-141. SCNT Control Field (C31:C0) .....	1038
19-142. SCNT Data Field (D31:D0) .....	1038
19-143. SHFT Program Field (P31:P0) .....	1040
19-144. SHFT Control Field (C31:C0).....	1040
19-145. SHFT Data Field (D31:D0).....	1040
19-146. WCAP Program Field (P31:P0) .....	1043
19-147. WCAP Control Field (C31:C0).....	1043
19-148. WCAP Data Field (D31:D0).....	1043
19-149. WCAPE Program Field (P31:P0) .....	1045

19-150. WCAPE Control Field (C31:C0).....	1045
19-151. WCAPE Data Field (D31:D0).....	1045
20-1. System Block Diagram.....	1049
20-2. HTU Block Diagram .....	1050
20-3. Example of a HTU Transfer .....	1050
20-4. Single Buffer Timing and Memory Representation.....	1052
20-5. Timing Example for Circular Buffer Mode .....	1053
20-6. Dual Buffer Timing .....	1054
20-7. Timing Example for Auto Switch Buffer Mode .....	1055
20-8. Timing for Disabling Control Packets .....	1056
20-9. Timing Example Including Lost Requests.....	1057
20-10. Timing, which Generates No Request Lost Error .....	1058
20-11. Timing, which Generates a Request Lost Error .....	1058
20-12. Timing Example for Two WCAP Instructions .....	1059
20-13. Timing of the WCAP, ECNT, PCNT Example .....	1062
20-14. Global Control Register (HTU GC) [offset = 00].....	1066
20-15. Control Packet Enable Register (HTU CPENA) [offset = 04h] .....	1067
20-16. Control Packet (CP) Busy Register 0 (HTU BUSY0) [offset = 08h] .....	1068
20-17. Control Packet (CP) Busy Register 1 (HTU BUSY1) [offset = 0Ch] .....	1069
20-18. Control Packet (CP) Busy Register 2 (HTU BUSY2) [offset = 10h] .....	1070
20-19. Control Packet (CP) Busy Register 3 (HTU BUSY3) [offset = 14h] .....	1071
20-20. Active Control Packet and Error Register (HTU ACPE) [offset = 18h] .....	1072
20-21. Request Lost and Bus Error Control Register (HTU RLBECTRL) [offset = 20h] .....	1074
20-22. Buffer Full Interrupt Enable Set Register (HTU BFINTS) [offset = 24h].....	1075
20-23. Buffer Full Interrupt Enable Clear Register (HTU BFINTC) [offset = 28h] .....	1076
20-24. Interrupt Mapping Register (HTU INTMAP) [offset = 2Ch].....	1077
20-25. Interrupt Offset Register 0 (HTU INTOFF0) [offset = 34h] .....	1078
20-26. Interrupt Offset Register 1 (HTU INTOFF1) [offset = 38h].....	1080
20-27. Buffer Initialization Mode Register (HTU BIM) [offset = 3Ch].....	1081
20-28. Request Lost Flag Register (HTU RLOSTFL) [offset = 40h] .....	1083
20-29. Buffer Full Interrupt Flag Register (HTU BFINTFL) [offset = 44h] .....	1084
20-30. BER Interrupt Flag Register (HTU BERINTFL) [offset = 48h] .....	1085
20-31. Memory Protection 1 Start Address Register (HTU MP1S) [offset = 4Ch].....	1086
20-32. Memory Protection 1 End Address Register (HTU MP1E) [offset = 50h] .....	1087
20-33. Debug Control Register (HTU DCTRL) [offset = 54h] .....	1088
20-34. Watch Point Register (HTU WPR) [offset = 58h].....	1089
20-35. Watch Mask Register (HTU WMR) [offset = 5Ch] .....	1090
20-36. Module Identification Register (HTU ID) [offset = 60h] .....	1091
20-37. Parity Control Register (HTU PCR) [offset = 64h] .....	1092
20-38. Parity Address Register (HTU PAR) [offset = 68h] .....	1093
20-39. Memory Protection Control and Status Register (HTU MPCS) [offset = 70h].....	1094
20-40. Memory Protection Start Address Register 0 (HTU MP0S) [offset = 74h] .....	1096
20-41. Memory Protection End Address Register (HTU MP0E) [offset = 78h] .....	1097
20-42. Initial Full Address A Register (HTU IFADDRA).....	1099
20-43. Initial Full Address B Register (HTU IFADDRB).....	1100
20-44. Initial NHET Address and Control Register (HTU IHADDRCT) .....	1101
20-45. Initial Transfer Count Register (HTU ITCOUNT) .....	1103
20-46. Current Full Address A Register (HTU CFADDRA).....	1104
20-47. Current Full Address B Register (HTU CFADDRB).....	1105

20-48. Current Frame Count Register (HTU CF_COUNT) .....	1106
21-1. Device Level Overview .....	1110
21-2. I/O Function Quick Start Flow Chart .....	1111
21-3. External Interrupt Function Quick Start Flow Chart .....	1112
21-4. GIO Module Diagram .....	1113
21-5. GIO Port A Module Diagram .....	1113
21-6. GIO Block Diagram.....	1114
21-7. Communication With the Data Output Register .....	1115
21-8. External Interrupt Block.....	1117
21-9. High-Level-Interrupt-Handling Block .....	1119
21-10. Low-Level-Interrupt-Handling Block.....	1119
21-11. GIO Global Control Register (GIOGCR0) [offset = FFF7 BC00h] .....	1123
21-12. GIO Interrupt Detect Register (GIOINTDET) [offset = FFF7 BC08h].....	1124
21-13. GIO Interrupt Polarity Register (GIOPOL) [offset = FFF7 BC0Ch] .....	1125
21-14. GIO Interrupt Enable Set Register (GIOENASET) [offset = FFF7 BC10h] .....	1126
21-15. GIO Interrupt Enable Clear Register (GIOENACLR) [offset = FFF7 BC14h].....	1127
21-16. GIO Interrupt Priority Register (GIOLVLSET) [offset = FFF7 BC18h] .....	1128
21-17. GIO Interrupt Priority Register (GIOLVLCLR) [offset = FFF7 BC1Ch] .....	1129
21-18. GIO Interrupt Flag Register (GIOFLG) [offset = FFF7 BC20h].....	1130
21-19. GIO Offset A Register (GIOFFA) [offset = FFF7 BC24h] .....	1131
21-20. GIO Offset B Register (GIOFFB) [offset = FFF7 BC28h] .....	1132
21-21. GIO Emulation A Register (GIOEMUA) [offset = FFF7 BC2Ch] .....	1133
21-22. GIO Emulation B Register (GIOEMUB) [offset = FFF7 BC30h].....	1134
21-23. GIO Data Direction Registers [A-B][7:0] (GIODIR[A-B]) [offset = FFF7 BC34h, FFF7 BC54h].....	1135
21-24. GIO Data Input Registers [A-B][7:0] (GIODIN[A-B]) [offset = FFF7 BC38h, FFF7 BC58h].....	1136
21-25. GIO Data Output Registers [A-B][7:0] (GIODOUT[A-B][7:0]) [offset = FFF7 BC3Ch, FFF7BC5Ch] .....	1137
21-26. GIO Data Set Registers [A-B][7:0] (GIODSET[A-B][7:0]) [offset = FFF7 BC40h, FFF7 BC60h].....	1138
21-27. GIO Data Clear Registers [A-B][7:0] (GIODCLR[A-B][7:0]) [offset = FFF7 BC44h, FFF7 BC64h] .....	1139
21-28. GIO Open Drain Registers [A-B][7:0] (GIOPDR[A-B][7:0]) [offset = FFF7 BC48h, FFF7 BC68h].....	1140
21-29. GIO Pull Disable Registers [A-B][7:0] (GIOPULDIS[A-B][7:0]) [offset = FFF7 BC4Ch, FFF7 BC6Ch].....	1141
21-30. GIO Pull Select Registers [A-B][7:0] (GIOPSL[A-B][7:0]) [offset = FFF7 BC50h, FFF7 BC70h].....	1142
22-1. DCAN Modules Message Ram .....	1150
22-2. Block Diagram .....	1151
22-3. Bit Timing .....	1153
22-4. CAN Bit-timing Configuration .....	1159
22-5. Structure of a Message Object .....	1161
22-6. Message RAM Representation in Debug/Suspend Mode .....	1164
22-7. Message RAM Representation in RAM Direct Access Mode .....	1165
22-8. Data Transfer Between IF1 / IF2 Registers and Message RAM .....	1167
22-9. Initialization of a Transmit Object .....	1169
22-10. Initialization of a Single Receive Object for Data Frames .....	1169
22-11. Initialization of a Single Receive Object for Remote Frames .....	1170
22-12. CPU Handling of a FIFO Buffer (Interrupt Driven) .....	1175
22-13. CAN Interrupt Topology 1 .....	1179
22-14. CAN Interrupt Topology 2 .....	1179
22-15. Local Power Down Mode Flow Diagram .....	1182
22-16. CAN Core in Silent Mode .....	1184
22-17. CAN Core in Loop Back Mode .....	1185
22-18. CAN Core in External Loop Back Mode .....	1186

22-19. CAN Core in Loop Back Combined with Silent Mode .....	1186
22-20. CAN Control Register (DCAN CTL) [offset = 0] .....	1192
22-21. Error and Status Register (DCAN ES) [offset = 4h].....	1195
22-22. Error Counter Register (DCAN ERRC) [offset = 8h].....	1197
22-23. Bit Timing Register (DCAN BTR) [offset = Ch] .....	1198
22-24. Interrupt Register (DCAN INT) [offset = 10h] .....	1199
22-25. Test Register (DCAN TEST) [offset = 14h] .....	1200
22-26. Parity Error Code Register (DCAN PERR) [offset = 1Ch].....	1201
22-27. Auto-Bus-On Time Register (DCAN ABOTR) [offset = 80h].....	1202
22-28. Transmission Request X Register (DCAN TXRQ X) [offset = 84h].....	1203
22-29. Transmission Request 12 Register [offset = 88h].....	1204
22-30. Transmission Request 34 Register [offset = 8Ch] .....	1204
22-31. Transmission Request 56 Register [offset = 90h].....	1204
22-32. Transmission Request 78 Register [offset = 94h].....	1204
22-33. New Data X Register (DCAN NWDAT X) [offset = 98h] .....	1205
22-34. New Data 12 Register [offset = 9Ch] .....	1206
22-35. New Data 34 Register [offset = A0h] .....	1206
22-36. New Data 56 Register [offset = A4h] .....	1206
22-37. New Data 78 Register [offset = A8h] .....	1206
22-38. Interrupt Pending X Register (DCAN INTPND X) [offset = ACh].....	1207
22-39. Interrupt Pending 12 Register [offset = B0h].....	1208
22-40. Interrupt Pending 34 Register [offset = B4h].....	1208
22-41. Interrupt Pending 56 Register [offset = B8h].....	1208
22-42. Interrupt Pending 78 Register [offset = BCh] .....	1208
22-43. Message Valid X Register (DCAN MSGVAL X) [offset = C0h].....	1209
22-44. Message Valid 12 Register [offset = C4h] .....	1210
22-45. Message Valid 34 Register [offset = C8h] .....	1210
22-46. Message Valid 56 Register [offset = CCh] .....	1210
22-47. Message Valid 78 Register [offset = D0h] .....	1210
22-48. Interrupt Multiplexer 12 Register [offset = D8h] .....	1211
22-49. Interrupt Multiplexer 34 Register [offset = DCh] .....	1211
22-50. Interrupt Multiplexer 56 Register [offset = E0h] .....	1211
22-51. Interrupt Multiplexer 78 Register [offset = E4h] .....	1211
22-52. IF1 Command Registers (DCAN IF1CMD) [offset = 100h] .....	1212
22-53. IF1 Command Registers (CAN IF2CMD) [offset = 120h] .....	1213
22-54. IF1 Mask Register (DCAN IF1MSK) [offset = 104h] .....	1215
22-55. IF2 Mask Register (DCAN IF2MSK) [offset = 124h].....	1215
22-56. IF1 Arbitration Register (DCAN IF1ARB) [offset = 108h] .....	1216
22-57. IF2 Arbitration Register (DCAN IF2ARB) [offset = 128h] .....	1216
22-58. IF1 Message Control Register (DCAN IF1MCTL) [offset = 10Ch] .....	1218
22-59. IF2 Message Control Register (DCAN IF2MCTL) [offset = 12Ch] .....	1218
22-60. IF1 Data A Register (DCAN IF1DATA) [offset = 110h].....	1220
22-61. IF1 Data B Register (DCAN IF1DATB) [offset = 114h].....	1220
22-62. IF2 Data A Register (DCAN IF2DATA) [offset = 130h].....	1220
22-63. IF2 Data B Register (DCAN IF2DATB) [offset = 134h].....	1220
22-64. IF3 Observation Register (DCAN IF3OBS) [offset = 140h] .....	1221
22-65. IF3 Mask Register (DCAN IF3MSK) [offset = 144h].....	1223
22-66. IF3 Arbitration Register (DCAN IF3ARB) [offset = 148h] .....	1224
22-67. IF3 Message Control Register (DCAN IF3MCTL) [offset = 14Ch] .....	1225



22-68. IF3 Data A Register (DCAN IF3DATA) [offset = 150h] .....	1227
22-69. IF3 Data B Register (DCAN IF3DATB) [offset = 154h] .....	1227
22-70. IF3 Update Enable 12 Register [offset = 160h] .....	1228
22-71. IF3 Update Enable 34 Register [offset = 164h] .....	1228
22-72. IF3 Update Enable 56 Register [offset = 168h] .....	1228
22-73. IF3 Update Enable 78 Register [offset = 16Ch] .....	1228
22-74. CAN TX IO Control Register (DCAN TIOC) [offset = 1E0h] .....	1229
22-75. CAN RX IO Control Register (DCAN RIOC) [offset = 1E4h] .....	1231
23-1. SPI Functional Logic Diagram .....	1237
23-2. SPI Three-Pin Operation .....	1238
23-3. Operation with $\overline{\text{SPISCS}}$ .....	1239
23-4. Operation with $\overline{\text{SPIENA}}$ .....	1240
23-5. SPI Five-Pin Option with $\overline{\text{SPIENA}}$ and $\overline{\text{SPISCS}}$ .....	1241
23-6. Format for Transmitting an 8-Bit Word .....	1242
23-7. Format for Receiving an 8-Bit Word .....	1242
23-8. Clock Mode with Polarity = 0 and Phase = 0 .....	1243
23-9. Clock Mode with Polarity = 0 and Phase = 1 .....	1243
23-10. Clock Mode with Polarity = 1 and Phase = 0 .....	1243
23-11. Clock Mode with Polarity = 1 and Phase = 1 .....	1244
23-12. Five Bits per Character (5-Pin Option) .....	1244
23-13. Typical Diagram when a Buffer in Master is in CSHOLD Mode (SPI-SPI) .....	1246
23-14. Block Diagram Shift Register, MSB First .....	1248
23-15. Block Diagram Shift Register, LSB First .....	1248
23-16. 2-data Line Mode (Phase 0, Polarity 0) .....	1251
23-17. Two-Pin Parallel Mode Timing Diagram (Phase 0, Polarity 0) .....	1251
23-18. 4-Data Line Mode (Phase 0, Polarity 0) .....	1252
23-19. 4 Pins Parallel Mode Timing Diagram (Phase 0, Polarity 0) .....	1252
23-20. Eight-data Line Mode (Phase 0, Polarity 0) .....	1253
23-21. 8 Pins Parallel Mode Timing Diagram (Phase 0, Polarity 0) .....	1254
23-22. I/O Paths during I/O Loopback Modes .....	1256
23-23. TG Interrupt Structure .....	1260
23-24. SPIFLG Interrupt Structure .....	1260
23-25. DMA Channel and Request Line (Logical) Structure in Multibuffer Mode .....	1261
23-26. SPI Global Control Register 0 (SPIGCR0) [offset = 00] .....	1266
23-27. SPI Global Control Register 1 (SPIGCR1) [offset = 04h] .....	1267
23-28. SPI Interrupt Register (SPIINT0) [offset = 08h] .....	1269
23-29. SPI Interrupt Level Register (SPILVL) [offset = 0Ch] .....	1271
23-30. SPI Flag Register (SPIFLG) [offset = 10h] .....	1272
23-31. SPI Pin Control Register 0 (SPIPC0) [offset = 14h] .....	1275
23-32. SPI Pin Control Register 1 (SPIPC1) [offset = 18h] .....	1277
23-33. SPI Pin Control Register 2 (SPIPC2) [offset = 1Ch] .....	1279
23-34. SPI Pin Control Register 3 (SPIPC3) [offset = 20h] .....	1280
23-35. SPI Pin Control Register 4 (SPIPC4) [offset = 24h] .....	1282
23-36. SPI Pin Control Register 5 (SPIPC5) [offset = 28h] .....	1284
23-37. SPI Pin Control Register 6 (SPIPC6) [offset = 2Ch] .....	1286
23-38. SPI Pin Control Register 7 (SPIPC7) [offset = 30h] .....	1288
23-39. SPI Pin Control Register 8 (SPIPC8) [offset = 34h] .....	1290
23-40. SPI Transmit Data Register 0 (SPIDAT0) [offset = 38h] .....	1292
23-41. SPI Transmit Data Register 1 (SPIDAT1) [offset = 3Ch] .....	1293

23-42. SPI Receive Buffer Register (SPIBUF) [offset = 40h] .....	1294
23-43. SPI Emulation Register (SPIEMU) [offset = 44h] .....	1296
23-44. SPI Delay Register (SPIDELAY) [offset = 48h].....	1297
23-45. Example: $t_{C2TDELAY} = 8$ VCLK Cycles .....	1298
23-46. Example: $t_{T2CDELAY} = 4$ VCLK Cycles .....	1298
23-47. Transmit-Data-Finished-to-ENA-Inactive-Timeout .....	1299
23-48. Chip-Select-Active-to-ENA-Signal-Active-Timeout .....	1299
23-49. SPI Default Chip Select Register (SPIDEF) [offset = 4Ch] .....	1300
23-50. SPI Data Format Registers (SPIFMT[3:0]) [offset = 5Ch-50h] .....	1301
23-51. Interrupt Vector 0 (INTVECT0) [offset = 60h] .....	1303
23-52. Interrupt Vector 1 (INTVECT1) [offset = 64h].....	1305
23-53. Parallel/Modulo Mode Control Register (SPIPMCTRL) [offset = 6Ch] .....	1307
23-54. Multi-buffer Mode Enable Register (MIBSPIE) [offset = 70h].....	1310
23-55. TG Interrupt Enable Set Register (TGITENST) [offset = 74h].....	1311
23-56. TG Interrupt Enable Clear Register (TGITENCR) [offset = 78h] .....	1312
23-57. Transfer Group Interrupt Level Set Register (TGITLVST) [offset = 7Ch] .....	1313
23-58. Transfer Group Interrupt Level Clear Register (TGITLVCR) [offset = 80h].....	1314
23-59. Transfer Group Interrupt Flag Register (TGINTFLAG) [offset = 84h] .....	1315
23-60. Tick Counter Operation .....	1316
23-61. Tick Count Register (TICKCNT) [offset = 90h] .....	1316
23-62. Last TG End Pointer (LTGPEND) [offset = 94h] .....	1317
23-63. MibSPI TG Control Registers (TGxCTRL) [offsets = 98h-D4h] .....	1318
23-64. DMA Channel Control Register (DMAxCTRL) [offset = D8h-F4h] .....	1321
23-65. DMAxCOUNT Register (ICOUNT) [offset = F8h-114h].....	1323
23-66. DMA Large Count Register (DMACNTLEN) [offset = 118h].....	1324
23-67. Multi-buffer RAM Uncorrectable Parity Error Control Register (UERRCTRL) [offset = 120h].....	1325
23-68. Multi-buffer RAM Uncorrectable Parity Error Status Register (UERRSTAT) [offset = 124h] .....	1326
23-69. RXRAM Uncorrectable Parity Error Address Register (UERRADDR1) [offset = 128h] .....	1327
23-70. TXRAM Uncorrectable Parity Error Address Register (UERRADDR0) [offset = 12Ch].....	1328
23-71. RXRAM Overrun Buffer Address Register (RXOVRN_BUF_ADDR) [offset = 130h].....	1329
23-72. I/O-Loopback Test Control Register (IOLPBKTSTCR) [offset = 134h] .....	1330
23-73. Multi-Buffer RAM Configuration .....	1332
23-74. Multi-buffer RAM Transmit Data Register [offset = Base + 000-1FFh].....	1335
23-75. Multi-buffer RAM Receive Buffer Register [offset = RAM Base + 200-3FFh] .....	1337
23-76. Memory Map for Parity Locations during Normal and Test Mode .....	1340
23-77. Example of Memory Mapped Parity Locations during Test Mode .....	1341
23-78. SPI/MibSPI Pins during Master mode 3-pin configuration .....	1342
23-79. SPI/MibSPI pins during Master mode 4-pin with $\overline{\text{SPITCS}}$ configuration. ....	1342
23-80. SPI/MibSPI pins during Master mode in 4pin with $\overline{\text{SPIENA}}$ configuration .....	1343
23-81. SPI/MibSPI pins during Master/Slave mode with 5-pin configuration .....	1343
23-82. SPI/MibSPI pins during Slave mode 3-pin configuration .....	1343
23-83. SPI/MibSPI pins during Slave mode in 4pin with $\overline{\text{SPIENA}}$ configuration .....	1344
23-84. SPI/MibSPI pins during Slave mode in 5pin configuration - (Single Slave). ....	1344
23-85. SPI/MibSPI pins during Slave mode in 5pin configuration - (Single/Multi Slave). ....	1344
24-1. Detailed SCI Block Diagram .....	1351
24-2. SCI/LIN Block Diagram .....	1352
24-3. Typical SCI Data Frame Formats .....	1353
24-4. Asynchronous Communication Bit Timing .....	1354
24-5. Superfractional Divider Example .....	1356

24-6. Idle-Line Multiprocessor Communication Format .....	1358
24-7. Address-Bit Multiprocessor Communication Format .....	1358
24-8. Receive Buffers .....	1359
24-9. Transmit Buffers .....	1360
24-10. General Interrupt Scheme .....	1361
24-11. Interrupt Generation for Given Flags .....	1362
24-12. LIN Protocol Message Frame Format: Master Header and Slave Response .....	1369
24-13. Header 3 Fields: Synch Break, Synch, and ID .....	1370
24-14. Response Format of LIN Message Frame .....	1370
24-15. Message Header in Terms of $T_{bit}$ .....	1373
24-16. ID Field .....	1373
24-17. Measurements for Synchronization .....	1375
24-18. Synchronization Validation Process and Baud Rate Adjustment .....	1376
24-19. Optional Embedded Checksum in Response for Extended Frames .....	1377
24-20. Checksum Compare and Send for Extended Frames .....	1378
24-21. TXRX Error Detector .....	1380
24-22. Classic Checksum Generation at Transmitting Node .....	1381
24-23. LIN 2.0-Compliant Checksum Generation at Transmitting Node .....	1381
24-24. ID Reception, Filtering and Validation .....	1383
24-25. LIN Message Frame Showing LIN Interrupt Timing and Sequence .....	1385
24-26. Wakeup Signal Generation .....	1389
24-27. SCI Global Control Register 0 (SCIGCR0) [offset = 00] .....	1393
24-28. SCI Global Control Register 1 (SCIGCR1) [offset = 04h] .....	1394
24-29. SCI Global Control Register 2 (SCIGCR2) [offset = 08h] .....	1398
24-30. SCI Set Interrupt Register (SCISSETINT) [offset = 0Ch] .....	1400
24-31. SCI Clear Interrupt Register (SCICLEARINT) [offset = 10h] .....	1403
24-32. SCI Set Interrupt Level Register (SCISSETINTLVL) [offset = 14h] .....	1407
24-33. SCI Clear Interrupt Level Register (SCICLEARINTLVL) [offset = 18h] .....	1410
24-34. SCI Flags Register (SCIFLR) [offset = 1Ch] .....	1413
24-35. SCI Interrupt Vector Offset 0 (SCIINTVECT0) [offset = 20h] .....	1420
24-36. SCI Interrupt Vector Offset 1 (SCIINTVECT1) [offset = 24h] .....	1421
24-37. SCI Format Control Register (SCIFORMAT) [offset = 28h] .....	1422
24-38. Baud Rate Selection Register (BRS) [offset = 2Ch] .....	1423
24-39. Receiver Emulation Data Buffer (SCIED) [offset = 30h] .....	1425
24-40. Receiver Data Buffer (SCIRD) [offset = 34h] .....	1426
24-41. Transmit Data Buffer Register (SCITD) [offset = 38h] .....	1427
24-42. SCI Pin I/O Control Register 0 (SCIPIO0) [offset = 3Ch] .....	1428
24-43. SCI Pin I/O Control Register 1 (SCIPIO1) [offset = 40h] .....	1429
24-44. SCI Pin I/O Control Register 2 (SCIPIO2) [offset = 44h] .....	1430
24-45. SCI Pin I/O Control Register 3 (SCIPIO3) [offset = 48h] .....	1431
24-46. SCI Pin I/O Control Register 4 (SCIPIO4) [offset = 4Ch] .....	1432
24-47. SCI Pin I/O Control Register 5 (SCIPIO5) [offset = 50h] .....	1433
24-48. SCI Pin I/O Control Register 6 (SCIPIO6) [offset = 54h] .....	1434
24-49. SCI Pin I/O Control Register 7 (SCIPIO7) [offset = 58h] .....	1435
24-50. SCI Pin I/O Control Register 8 (SCIPIO8) [offset = 5Ch] .....	1436
24-51. LIN Compare Register (LINCOMPARE) [offset = 60h] .....	1437
24-52. LIN Receive Buffer 0 Register (LINRD0) [offset = 64h] .....	1438
24-53. LIN Receive Buffer 1 Register (RD1) [offset = 68h] .....	1439
24-54. LIN Mask Register (LINMASK) [offset = 6Ch] .....	1440

24-55. LIN Identification Register (LINID) [offset = 70h] .....	1441
24-56. LIN Transmit Buffer 0 Register (LINTD0) [offset = 74h] .....	1442
24-57. LIN Transmit Buffer 1 Register (LINTD1) [offset = 78h] .....	1443
24-58. Maximum Baud Rate Selection Register (MBRS) [offset = 7Ch] .....	1444
24-59. Input/Output Error Enable Register (IODFTCTRL) [offset = 90h] .....	1445
24-60. GPIO Functionality .....	1447
25-1. Detailed SCI Block Diagram .....	1451
25-2. Typical SCI Data Frame Formats .....	1452
25-3. Asynchronous Communication Bit Timing .....	1453
25-4. Idle-Line Multiprocessor Communication Format .....	1455
25-5. Address-Bit Multiprocessor Communication Format .....	1456
25-6. General Interrupt Scheme .....	1457
25-7. Interrupt Generation for Given Flags .....	1458
25-8. SCI Global Control Register 0 (SCIGCR0) [offset = 00] .....	1466
25-9. SCI Global Control Register 1 (SCIGCR1) [offset = 04h] .....	1467
25-10. SCI Set Interrupt Register (SCISSETINT) [offset = 0Ch] .....	1470
25-11. SCI Clear Interrupt Register (SCICLEARINT) [offset = 10h] .....	1472
25-12. SCI Set Interrupt Level Register (SCISSETINTLVL) [offset = 14h] .....	1474
25-13. SCI Clear Interrupt Level Register (SCICLEARINTLVL) [offset = 18h] .....	1476
25-14. SCI Flags Register (SCIFLR) [offset = 1Ch] .....	1478
25-15. SCI Interrupt Vector Offset 0 (SCIINTVECT0) [offset = 20h] .....	1482
25-16. SCI Interrupt Vector Offset 1 (SCIINTVECT1) [offset = 24h] .....	1483
25-17. SCI Format Control Register (SCIFORMAT) [offset = 28h] .....	1484
25-18. Baud Rate Selection Register (BRS) [offset = 2Ch] .....	1485
25-19. Receiver Emulation Data Buffer (SCIED) [offset = 30h] .....	1487
25-20. Receiver Data Buffer (SCIRD) [offset = 34h] .....	1488
25-21. Transmit Data Buffer Register (SCITD) [offset = 38h] .....	1489
25-22. SCI Pin I/O Control Register 0 (SCIPIO0) [offset = 3Ch] .....	1490
25-23. SCI Pin I/O Control Register 1 (SCIPIO1) [offset = 40h] .....	1491
25-24. SCI Pin I/O Control Register 2 (SCIPIO2) [offset = 44h] .....	1492
25-25. SCI Pin I/O Control Register 3 (SCIPIO3) [offset = 48h] .....	1493
25-26. SCI Pin I/O Control Register 4 (SCIPIO4) [offset = 4Ch] .....	1494
25-27. SCI Pin I/O Control Register 5 (SCIPIO5) [offset = 50h] .....	1495
25-28. SCI Pin I/O Control Register 6 (SCIPIO6) [offset = 54h] .....	1496
25-29. SCI Pin I/O Control Register 7 (SCIPIO7) [offset = 58h] .....	1497
25-30. SCI Pin I/O Control Register 8 (SCIPIO8) [offset = 5Ch] .....	1498
25-31. Input/Output Error Enable Register (IODFTCTRL) [offset = 90h] .....	1499
25-32. GPIO Functionality .....	1501
26-1. Multiple I2C Modules Connection Diagram .....	1504
26-2. Simple I2C Block Diagram .....	1506
26-3. Clocking Diagram for the I2C Module .....	1507
26-4. Bit Transfer on the I2C Bus .....	1509
26-5. I2C Module START and STOP Conditions .....	1509
26-6. I2C Module Data Transfer .....	1510
26-7. I2C Module 7-Bit Addressing Format .....	1510
26-8. I2C Module 10-bit Addressing Format .....	1511
26-9. I2C Module 7-Bit Addressing Format with Repeated START .....	1511
26-10. I2C Module in Free Data Format .....	1511
26-11. Arbitration Procedure Between Two Master Transmitters .....	1515

26-12. Synchronization of Two I2C Clock Generators During Arbitration .....	1516
26-13. I2C Own Address Manager Register (I2COAR) [Offset = 00] .....	1521
26-14. I2C Interrupt Mask Register (I2CIMR) [Offset = 04h] .....	1522
26-15. I2C Status Register (I2CSR) [Offset = 08h] .....	1523
26-16. I2C Clock Divider Low Register (I2CCKL) [Offset = 0Ch] .....	1526
26-17. I2C Clock Control High Register (I2CCKH) [Offset = 10h] .....	1527
26-18. I2C Data Count Register (I2CCNT) [Offset = 14h] .....	1528
26-19. I2C Data Receive Register (I2CDRR) [Offset = 18h] .....	1529
26-20. I2C Slave Address Register (I2CSAR) [Offset = 1Ch] .....	1530
26-21. I2C Data Transmit Register (I2CDXR) [Offset = 20h] .....	1531
26-22. I2C Mode Register (I2CMDR) [Offset = 24h] .....	1532
26-23. Typical Timing Diagram of Repeat Mode .....	1534
26-24. I2C Interrupt Vector Register (I2CIVR) [Offset = 28h] .....	1535
26-25. I2C Extended Mode Register (I2CEMDR) [Offset = 2Ch] .....	1536
26-26. I2C Prescale Register (I2CPSC) [Offset = 30h] .....	1537
26-27. I2C Peripheral ID Register 1 (I2CPID1) [Offset = 34h] .....	1538
26-28. I2C Peripheral ID Register 2 (I2CPID2) [Offset = 38h] .....	1539
26-29. I2C DMA Control Register (I2CDMACR) [Offset = 3Ch] .....	1540
26-30. I2C Pin Function Register (I2CPFNC) [Offset = 48h] .....	1541
26-31. I2C Pin Direction Register (I2CPDIR) [Offset = 4Ch] .....	1542
26-32. I2C Data Input Register (I2CDIN) [Offset = 50h] .....	1543
26-33. I2C Data Output Register (I2CDOUT) [Offset 0x54] .....	1544
26-34. I2C Data Set Register (I2CDSET) [Offset = 58h] .....	1545
26-35. I2C Data Clear Register (I2CDCLR) [Offset = 5Ch] .....	1546
26-36. I2C Pin Open Drain Register (I2CPDR) [Offset = 60h] .....	1547
26-37. I2C Pull Disable Register (I2CPDIS) [Offset = 64h] .....	1548
26-38. I2C Pull Disable Register (I2CPDIS) [Offset = 68h] .....	1549
26-39. I2C Pins Slew Rate Select Register (I2CSRS) [Offset = 6Ch] .....	1550
26-40. Difference between Normal Operation and Backward Compatibility Mode .....	1551
27-1. EMAC and MDIO Block Diagram.....	1555
27-2. Ethernet Configuration—MII Connections.....	1558
27-3. Ethernet Configuration—RMII Connections.....	1559
27-4. Ethernet Frame Format.....	1560
27-5. Basic Descriptor Format.....	1562
27-6. Typical Descriptor Linked List.....	1563
27-7. Transmit Buffer Descriptor Format.....	1566
27-8. Receive Buffer Descriptor Format.....	1569
27-9. EMAC Control Module Block Diagram.....	1573
27-10. MDIO Module Block Diagram.....	1575
27-11. EMAC Module Block Diagram .....	1579
27-12. EMAC Control Module Revision ID Register (REVID).....	1601
27-13. EMAC Control Module Software Reset Register (SOFTRESET) .....	1602
27-14. EMAC Control Module Interrupt Control Register (INTCONTROL).....	1603
27-15. EMAC Control Module Receive Threshold Interrupt Enable Register (C0RXTHRESHEN) .....	1604
27-16. EMAC Control Module Receive Interrupt Enable Register (C0RXEN) .....	1605
27-17. EMAC Control Module Transmit Interrupt Enable Register (C0TXEN).....	1606
27-18. EMAC Control Module Miscellaneous Interrupt Enable Register (C0MISCEN).....	1607
27-19. EMAC Control Module Receive Threshold Interrupt Status Register (C0RXTHRESHSTAT).....	1608
27-20. EMAC Control Module Receive Interrupt Status Register (C0RXSTAT).....	1609



27-21. EMAC Control Module Transmit Interrupt Status Register (C0TXSTAT) .....	1610
27-22. EMAC Control Module Miscellaneous Interrupt Status Register (COMISCSTAT) .....	1611
27-23. EMAC Control Module Receive Interrupts Per Millisecond Register (C0RXIMAX) .....	1612
27-24. EMAC Control Module Transmit Interrupts Per Millisecond Register (C0TXIMAX) .....	1613
27-25. MDIO Revision ID Register (REVID).....	1615
27-26. MDIO Control Register (CONTROL).....	1616
27-27. PHY Acknowledge Status Register (ALIVE) .....	1617
27-28. PHY Link Status Register (LINK).....	1618
27-29. MDIO Link Status Change Interrupt (Unmasked) Register (LINKINTRAW) .....	1619
27-30. MDIO Link Status Change Interrupt (Masked) Register (LINKINTMASKED) .....	1620
27-31. MDIO User Command Complete Interrupt (Unmasked) Register (USERINTRAW).....	1621
27-32. MDIO User Command Complete Interrupt (Masked) Register (USERINTMASKED) .....	1622
27-33. MDIO User Command Complete Interrupt Mask Set Register (USERINTMASKSET).....	1623
27-34. MDIO User Command Complete Interrupt Mask Clear Register (USERINTMASKCLEAR) .....	1624
27-35. MDIO User Access Register 0 (USERACCESS0) .....	1625
27-36. MDIO User PHY Select Register 0 (USERPHYSEL0) .....	1626
27-37. MDIO User Access Register 1 (USERACCESS1) .....	1627
27-38. MDIO User PHY Select Register 1 (USERPHYSEL1) .....	1628
27-39. Transmit Revision ID Register (TXREVID).....	1633
27-40. Transmit Control Register (TXCONTROL).....	1634
27-41. Transmit Teardown Register (TXTEARDOWN) .....	1635
27-42. Receive Revision ID Register (RXREVID) .....	1636
27-43. Receive Control Register (RXCONTROL) .....	1637
27-44. Receive Teardown Register (RXTEARDOWN) .....	1638
27-45. Transmit Interrupt Status (Unmasked) Register (TXINTSTATRAW) .....	1639
27-46. Transmit Interrupt Status (Masked) Register (TXINTSTATMASKED).....	1640
27-47. Transmit Interrupt Mask Set Register (TXINTMASKSET) .....	1641
27-48. Transmit Interrupt Mask Clear Register (TXINTMASKCLEAR) .....	1642
27-49. MAC Input Vector Register (MACINVECTOR) .....	1643
27-50. MAC End Of Interrupt Vector Register (MACEOIVECTOR).....	1644
27-51. Receive Interrupt Status (Unmasked) Register (RXINTSTATRAW).....	1645
27-52. Receive Interrupt Status (Masked) Register (RXINTSTATMASKED) .....	1646
27-53. Receive Interrupt Mask Set Register (RXINTMASKSET).....	1647
27-54. Receive Interrupt Mask Clear Register (RXINTMASKCLEAR).....	1648
27-55. MAC Interrupt Status (Unmasked) Register (MACINTSTATRAW) .....	1649
27-56. MAC Interrupt Status (Masked) Register (MACINTSTATMASKED).....	1650
27-57. MAC Interrupt Mask Set Register (MACINTMASKSET) .....	1651
27-58. MAC Interrupt Mask Clear Register (MACINTMASKCLEAR) .....	1652
27-59. Receive Multicast/Broadcast/Promiscuous Channel Enable Register (RXMBPENABLE).....	1653
27-60. Receive Unicast Enable Set Register (RXUNICASTSET) .....	1656
27-61. Receive Unicast Clear Register (RXUNICASTCLEAR) .....	1657
27-62. Receive Maximum Length Register (RXMAXLEN) .....	1658
27-63. Receive Buffer Offset Register (RXBUFFEROFFSET).....	1659
27-64. Receive Filter Low Priority Frame Threshold Register (RXFILTERLOWTHRESH) .....	1660
27-65. Receive Channel <i>n</i> Flow Control Threshold Register (RX <i>n</i> FLOWTHRESH).....	1661
27-66. Receive Channel <i>n</i> Free Buffer Count Register (RX <i>n</i> FREEBUFFER) .....	1662
27-67. MAC Control Register (MACCONTROL).....	1663
27-68. MAC Status Register (MACSTATUS) .....	1665
27-69. Emulation Control Register (EMCONTROL).....	1667



27-70. FIFO Control Register (FIFOCONTROL).....	1668
27-71. MAC Configuration Register (MACCONFIG) .....	1669
27-72. Soft Reset Register (SOFTRESET).....	1670
27-73. MAC Source Address Low Bytes Register (MACSRCADDRLO) .....	1671
27-74. MAC Source Address High Bytes Register (MACSRCADDRHI) .....	1672
27-75. MAC Hash Address Register 1 (MACHASH1) .....	1673
27-76. MAC Hash Address Register 2 (MACHASH2) .....	1674
27-77. Back Off Random Number Generator Test Register (BOFFTEST).....	1675
27-78. Transmit Pacing Algorithm Test Register (TPACETEST) .....	1676
27-79. Receive Pause Timer Register (RXPAUSE).....	1677
27-80. Transmit Pause Timer Register (TXPAUSE) .....	1678
27-81. MAC Address Low Bytes Register (MACADDRLO) .....	1679
27-82. MAC Address High Bytes Register (MACADDRHI).....	1680
27-83. MAC Index Register (MACINDEX) .....	1681
27-84. Transmit Channel <i>n</i> DMA Head Descriptor Pointer Register (TX <i>n</i> HDP).....	1682
27-85. Receive Channel <i>n</i> DMA Head Descriptor Pointer Register (RX <i>n</i> HDP) .....	1683
27-86. Transmit Channel <i>n</i> Completion Pointer Register (TX <i>n</i> CP) .....	1684
27-87. Receive Channel <i>n</i> Completion Pointer Register (RX <i>n</i> CP).....	1685
27-88. Statistics Register .....	1686
28-1. OHCI Revision Number Register (HCREVISION) [address = FCF78B00h].....	1700
28-2. HC Operating Mode Register (HCCONTROL) [address = FCF78B04h].....	1701
28-3. HC Command and Status Register (HCCOMMANDSTATUS) [address = FCF78B08h] .....	1702
28-4. HC Interrupt Status Register (HCINTERRUPTSTATUS) [address = FCF78B0Ch] .....	1703
28-5. HC Interrupt Enable Register (HCINTERRUPTENABLE) [address = FCF78B10h].....	1704
28-6. HC Interrupt Disable Register (HCINTERRUPTDISABLE) [address = FCF78B14h].....	1705
28-7. HC HCCA Address Register (HCHCCA) [address = FCF78B18h] .....	1706
28-8. HC Current Periodic Register (HCPERIODCURRENTED) [address = FCF78B1Ch] .....	1706
28-9. HC Head Control Register (HCCONTROLHEADED) [address = FCF78B20h].....	1707
28-10. HC Current Control Register (HCCONTROLCURRENTED) [address = FCF78B24h].....	1707
28-11. HC Head Bulk Register (HCBULKHEADED) [address = FCF78B28h] .....	1708
28-12. HC Current Bulk Register (HCBULKCURRENTED) [address = FCF78B2Ch].....	1708
28-13. HC Head Done Register (HCDONEHEAD) [address = FCF78B30h] .....	1709
28-14. HC Frame Interval Register (HCFMINTERVAL) [address = FCF78B34h].....	1709
28-15. HC Frame Remaining Register (HCFMREMAINING) [address = FCF78B38h].....	1710
28-16. HC Frame Number Register (HCFMNUMBER) [address = FCF78B3Ch] .....	1710
28-17. HC Periodic Start Register (HCPERIODICSTART) [address = FCF78B40h] .....	1711
28-18. HC Low-Speed Threshold Register (HCLSTHRESHOLD) [address = FCF78B44h] .....	1711
28-19. HC Root Hub A Register (HCRHDESCRIPTORA) [address = FCF78B48h].....	1712
28-20. HC Root Hub B Register (HCRHDESCRIPTORB) [address = FCF78B4Ch] .....	1713
28-21. HC Root Hub Status Register (HCRHSTATUS) [address = FCF78B50h].....	1714
28-22. HC Port 0 Status and Control Register (HCRHPORTSTATUS0) [address = FCF78B54h] .....	1715
28-23. HC Port 1 Status and Control Register (HCRHPORTSTATUS1) [address = FCF78B58h] .....	1717
28-24. Host UE Address Register (HOSTUEADDR) [address = FCF78BE0h].....	1719
28-25. Host UE Status Register (HOSTUESTATUS) [address = FCF78BE4h] .....	1719
28-26. Host Time-out Control Register (HOSTTIMEOUTCTRL) [address = FCF78BE8h] .....	1720
28-27. Host Revision Register (HOSTREVISION) [address = FCF78BECh] .....	1720
28-28. Revision Register (REV) [address = FCF78A00h].....	1724
28-29. Endpoint Selection Register (EP_NUM) [address = FCF78A02h] .....	1724
28-30. Data Register (DATA) [address = FCF78A04h] .....	1725

28-31. Control Register (CTRL) [address = FCF78A06h] .....	1726
28-32. Status Register (STAT_FLG) [address = FCF78A08h].....	1727
28-33. Receive FIFO Status Register (RXFSTAT) [address = FCF78A0Ah] .....	1729
28-34. System Configuration Register 1 (SYSCON1) [address = FCF78A0Ch] .....	1730
28-35. System Configuration Register 2 (SYSCON2) [address = FCF78A0Eh].....	1731
28-36. Device Status Register (DEVSTAT) [address = FCF78A10h].....	1732
28-37. Start of Frame Register (SOF) [address = FCF78A12h] .....	1734
28-38. Interrupt Enable Register (IRQ_EN) [address = FCF78A14h].....	1735
28-39. DMA Interrupt Enable Register (DMA_IRQ_EN) [address = FCF78A16h] .....	1736
28-40. Interrupt Source Register (IRQ_SRC) [address = FCF78A18h] .....	1736
28-41. Non-ISO Endpoint Interrupt Status Register (EPN_STAT) [address = FCF78A1Ah] .....	1738
28-42. Non-ISO DMA Interrupt Status Register (DMAN_STAT) [address = FCF78A1Ch] .....	1739
28-43. DMA Receive Channels Configuration Register (RXDMA_CFG) [address = FCF78A20h] .....	1740
28-44. DMA Transmit Channels Configuration Register (TXDMA_CFG) [address = FCF78A22h].....	1741
28-45. DMA FIFO Data Register (DATA_DMA) [address = FCF78A24h].....	1742
28-46. Transmit DMA Control Register n (TXDMA <sub>n</sub> ) [address = FCF78A28h — FCF78A2Ch] .....	1743
28-47. Receive DMA Control Register n (RXDMA <sub>n</sub> ) [address = FCF78A30h — FCF78A34h] .....	1744
28-48. Endpoint 0 Configuration Register (EP0) [address = FCF78A40h] .....	1745
28-49. Receive Endpoint n Configuration Register (EP <sub>n</sub> _RX) [address = FCF78A42h — FCF78A5Eh] .....	1745
28-50. Transmit Endpoint n Configuration Register (EP <sub>n</sub> _TX) [address = FCF78A62h — FCF78A7Eh].....	1747
28-51. Non-Isochronous, Non-Control OUT Transaction Phases and Interrupts.....	1750
28-52. Non-Isochronous IN Transaction Phases and Interrupts.....	1753
28-53. Isochronous OUT Transaction Phases and Interrupts.....	1756
28-54. Isochronous IN Transaction Phases and Interrupts.....	1757
28-55. Stages and Transaction Phases of Autodecoded Control Transfers .....	1759
28-56. Stages and Transaction Phases of Non-Autodecoded Control Transfers .....	1760
28-57. Example of RAM Organization .....	1767
28-58. Device Configuration Routine.....	1768
28-59. Endpoint Configuration Routine .....	1769
28-60. Prepare for USB RX Transfers Routine .....	1770
28-61. Prepare for TX Transfer on Endpoint n Routine.....	1771
28-62. General USB Interrupt ISR Source Parsing Flowchart .....	1773
28-63. Setup Interrupt Handler .....	1774
28-64. Parse Command Routine (Setup Stage Control Transfer Request).....	1775
28-65. Endpoint 0 RX Interrupt Handler .....	1776
28-66. Prepare for Control Write Status Stage Routine .....	1777
28-67. Endpoint 0 TX Interrupt Handler.....	1778
28-68. Prepare for Control Read Status Stage Routine .....	1779
28-69. USB Device Controller Device State Transitions.....	1781
28-70. Typical Operation for USB Device State Changed Interrupt Handler .....	1782
28-71. Attached/Unattached Handler.....	1783
28-72. Configuration Changed Handler .....	1784
28-73. Address Changed Handler .....	1785
28-74. USB Device Reset Handler Flowchart.....	1786
28-75. Typical Operation for USB Suspend/Resume General USB Interrupt Handler .....	1787
28-76. Non-ISO Endpoint-Specific (Except EP 0) ISR Flowchart .....	1788
28-77. Non-Isochronous Non-Control Endpoint Receive Interrupt Handler .....	1789
28-78. Read Non-Isochronous RX FIFO Data Flowchart.....	1790
28-79. Non-Isochronous Non-Control Endpoint Transmit Interrupt Handler .....	1791

28-80. Write Non-Isochronous TX FIFO Data Flowchart .....	1792
28-81. SOF Interrupt Handler Flowchart.....	1793
28-82. Read Isochronous RX FIFO Data Flowchart .....	1794
28-83. Write Isochronous TX FIFO Data Flowchart.....	1795
28-84. Non-ISO RX DMA Transaction Example (RX_TC=2) .....	1797
28-85. Non-ISO RX DMA Start Routine .....	1798
28-86. Non-ISO RX DMA EOT Interrupt Handler.....	1799
28-87. Non-ISO RX DMA Transactions Count Interrupt Handler.....	1800
28-88. ISO RX DMA Transaction .....	1800
28-89. ISO RX DMA Start Routine .....	1801
28-90. Non-ISO TX DMA Start Routine.....	1802
28-91. Non-ISO TX DMA Done Interrupt Handler .....	1803
28-92. ISO TX DMA Start Routine.....	1804
28-93. Power Management Signal Values .....	1806
28-94. Power Management Flowchart .....	1807
28-95. Pin Group 0 USB Host Using 6-Wire USB Transceiver .....	1810
28-96. USB Device Connections on USB Pin Group 0 Using 6-Wire Transceiver.....	1811
29-1. DMM Block Diagram .....	1814
29-2. Trace Mode Packet Format .....	1816
29-3. Direct Data Mode Packet Format .....	1816
29-4. Packet Sync Signal Example .....	1818
29-5. Example Single Packet Transmission .....	1818
29-6. Interrupt Structure .....	1819
29-7. DMM Global Control Register (DMMGLBCTRL) [offset = 00h] .....	1821
29-8. DMM Interrupt Set Register (DMMINTSET) [offset = 04h].....	1823
29-9. DMM Interrupt Clear Register (DMMINTCLR) [offset = 08h] .....	1827
29-10. DMM Interrupt Level Register (DMMINTLVL) [offset = 0Ch] .....	1832
29-11. DMM Interrupt Flag Register (DMMINTFLG) [offset = 10h].....	1834
29-12. DMM Interrupt Offset 1 Register (DMMOFF1) [offset = 14h] .....	1838
29-13. DMM Interrupt Offset 2 Register (DMMOFF2) [offset = 18h] .....	1839
29-14. DMM Direct Data Mode Destination Register (DMMDDMDEST) [offset = 1Ch] .....	1840
29-15. DMM Direct Data Mode Blocksize Register (DMMDDMBL) [offset = 20h] .....	1841
29-16. DMM Direct Data Mode Pointer Register (DMMDDMPT) [offset = 24h] .....	1842
29-17. DMM Direct Data Mode Interrupt Pointer Register (DMMINTPT) [offset = 28h] .....	1843
29-18. DMM Destination x Region 1 (DMMDESTxREG1) [offset = 2Ch, 3Ch, 4Ch, 5Ch].....	1844
29-19. DMM Destination x Blocksize 1 (DMMDESTxBL1) [offset = 30h, 40h, 50h, 60h] .....	1845
29-20. DMM Destination x Region 2 (DMMDESTxREG2) [offset = 34h, 44h, 54h, 64h].....	1846
29-21. DMM Destination x Blocksize 2 (DMMDESTxBL2) [offset = 38h, 48h, 58h, 68h] .....	1847
29-22. DMM Pin Control 0 (DMMPC0) [offset = 6Ch].....	1848
29-23. DMM Pin Control 1 (DMMPC1) [offset = 70h] .....	1850
29-24. DMM Pin Control 2 (DMMPC2) [offset = 74h] .....	1852
29-25. DMM Pin Control 3 (DMMPC3) [offset = 78h] .....	1853
29-26. DMM Pin Control 4 (DMMPC4) [offset = 7Ch].....	1855
29-27. DMM Pin Control 5 (DMMPC5) [offset = 80h] .....	1857
29-28. DMM Pin Control 6 (DMMPC6) [offset = 84h] .....	1859
29-29. DMM Pin Control 7 (DMMPC7) [offset = 88h] .....	1861
29-30. DMM Pin Control 8 (DMMPC8) [offset = 8Ch].....	1863
30-1. Block Diagram RAM Trace Port Module .....	1866
30-2. Packet Format Trace Mode for RAM Locations .....	1867

---

30-3. Packet Format Trace Mode for Peripheral Locations.....	1867
30-4. Packet Format in Direct Data Mode.....	1869
30-5. Example for Trace Region Setup.....	1870
30-6. FIFO Overflow Handling .....	1871
30-7. RTP Packet Transfer with Sync Signal .....	1872
30-8. Packet Format in Trace Mode .....	1872
30-9. RTP Global Control Register (RTPGLBCTRL) [offset = 00h].....	1875
30-10. RTP Trace Enable Register (RTPTRENA) [offset = 04h] .....	1878
30-11. RTP Global Status Register (RTPGSR) [offset = 08h] .....	1879
30-12. RTP RAM 1 Trace Region [1:2] Register (RTPRAM1REG[1:2]) [offset = 0Ch, 10h].....	1881
30-13. RTP RAM 2 Trace Region [1:2] Register (RTPRAM2REG[1:2]) [offset = 14h, 18h] .....	1882
30-14. RTP Peripheral Trace Region [1:2] Register (RTPPERREG[1:2]) [offset = 24h, 28h] .....	1883
30-15. RTP Direct Data Mode Write Register (RTPDDMW) [offset = 2Ch].....	1884
30-16. RTP Pin Control 0 Register (RTPPC0) [offset = 34h] .....	1885
30-17. RTP Pin Control 1 Register (RTPPC1) [offset = 38h] .....	1886
30-18. RTP Pin Control 2 Register (RTPPC2) [offset = 3Ch].....	1887
30-19. RTP Pin Control 3 Register (RTPPC3) [offset = 40h] .....	1888
30-20. RTP Pin Control 4 Register (RTPPC4) [offset = 44h] .....	1889
30-21. RTP Pin Control 5 Register (RTPPC5) [offset = 48h] .....	1890
30-22. RTP Pin Control 6 Register (RTPPC6) [offset = 4Ch].....	1891
30-23. RTP Pin Control 7 Register (RTPPC7) [offset = 50h] .....	1893
30-24. RTP Pin Control 8 Register (RTPPC8) [offset = 54h] .....	1894

## List of Tables

2-1.	Definition of Terms .....	93
2-2.	Bus Master / Slave Access Privileges .....	95
2-3.	32-bit Word-Invariant Big Endian Format .....	95
2-4.	Clock Sources.....	96
2-5.	Clock Domains.....	97
2-6.	Typical Low-Power Modes .....	99
2-7.	Clock Test Mode Options .....	100
2-8.	DCC1 Counter 0 Clock Inputs .....	102
2-9.	DCC1 Counter 1 Clock / Signal Inputs .....	102
2-10.	DCC2 Counter 0 Clock Inputs .....	102
2-11.	DCC2 Counter 1 Clock / Signal Inputs .....	102
2-12.	Module Registers / Memories Memory Map .....	104
2-13.	Flash Memory Banks and Sectors .....	108
2-14.	PBIST Memory Grouping .....	110
2-15.	PBIST Algorithm Mapping .....	111
2-16.	Memory Initialization Select Mapping .....	113
2-17.	Causes of Resets .....	115
2-18.	Primary System Control Registers Summary .....	118
2-19.	SYS Pin Control Register 1 (SYSPC1) Field Descriptions .....	120
2-20.	SYS Pin Control Register 2 (SYSPC2) Field Descriptions .....	121
2-21.	SYS Pin Control Register 3 (SYSPC3) Field Descriptions .....	122
2-22.	SYS Pin Control Register 4 (SYSPC4) Field Descriptions .....	123
2-23.	SYS Pin Control Register 5 (SYSPC5) Field Descriptions .....	124
2-24.	SYS Pin Control Register 6 (SYSPC6) Field Descriptions .....	125
2-25.	SYS Pin Control Register 7 (SYSPC7) Field Descriptions .....	126
2-26.	SYS Pin Control Register 8 (SYSPC8) Field Descriptions .....	127
2-27.	SYS Pin Control Register 9 (SYSPC9) Field Descriptions .....	128
2-28.	Clock Source Disable Register (CSDIS) Field Descriptions.....	129
2-29.	Clock Sources Table .....	129
2-30.	Clock Source Disable Set Register (CSDISSET) Field Descriptions .....	130
2-31.	Clock Source Disable Clear Register (CSDISCLR) Field Descriptions .....	131
2-32.	Clock Domain Disable Register (CDDIS) Field Descriptions.....	132
2-33.	Clock Domain Disable Set Register (CDDISSET) Field Descriptions.....	133
2-34.	Clock Domain Disable Clear Register (CDDISCLR) Field Descriptions .....	135
2-35.	GCLK, HCLK, VCLK, and VCLK2 Source Register (GHVSR) Field Descriptions.....	137
2-36.	Peripheral Asynchronous Clock Source Register (VCLKASRC) Field Descriptions .....	139
2-37.	RTI Clock Source Register (RCLKSRC) Field Descriptions.....	140
2-38.	Clock Source Valid Register (CSVSTAT) Field Descriptions .....	141
2-39.	Memory Self-Test Global Control Register (MSTGCR) Field Descriptions.....	142
2-40.	Memory Hardware Initialization Global Control Register (MINITGCR) Field Descriptions .....	143
2-41.	MBIST Controller/Memory Initialization Enable Register (MSINENA) Field Descriptions .....	144
2-42.	Memory Self-Test Fail Status Register (MSTFAIL) Field Descriptions .....	145
2-43.	MSTC Global Status Register (MSTCGSTAT) Field Descriptions .....	146
2-44.	Memory Hardware Initialization Status Register (MINISTAT) Field Descriptions .....	147
2-45.	PLL Control Register 1 (PLLCTL1) Field Descriptions.....	148
2-46.	PLL Control Register 2 (PLLCTL2) Field Descriptions.....	149
2-47.	SYS Pin Control Register 10 (SYSPC10) Field Descriptions .....	151

2-48.	Die Identification Register, Lower Word (DIEIDL) Field Descriptions .....	152
2-49.	Die Identification Register, Upper Word (DIEIDH) Field Descriptions .....	153
2-50.	LPO/Clock Monitor Control Register (LPOMONCTL) Field Descriptions .....	154
2-51.	Clock Test Register (CLKTEST) Field Descriptions .....	157
2-52.	DFT Control Register (DFTCTRLREG) Field Descriptions .....	159
2-53.	DFT Control Register 2 (DFTCTRLREG2) Field Descriptions .....	160
2-54.	General Purpose Register (GPREG1) Field Descriptions .....	161
2-55.	Imprecise Fault Status Register (IMPFSTATUS) Field Descriptions .....	162
2-56.	Imprecise Fault Write Address Register (IMPFTADD) Field Descriptions .....	164
2-57.	System Software Interrupt Request 1 Register (SSIR1) Field Descriptions .....	165
2-58.	System Software Interrupt Request 2 Register (SSIR2) Field Descriptions .....	166
2-59.	System Software Interrupt Request 3 Register (SSIR3) Field Descriptions .....	167
2-60.	System Software Interrupt Request 4 Register (SSIR4) Field Descriptions .....	168
2-61.	RAM Control Register (RAMGCR) Field Descriptions .....	169
2-62.	Bus Matrix Module Control Register 1 (BMMCR) Field Descriptions .....	170
2-63.	CPU Reset Control Register (CPURSTGCR) Field Descriptions .....	171
2-64.	Clock Control Register (CLKCNTRL) Field Descriptions .....	172
2-65.	ECP Control Register (ECPCNTL) Field Descriptions .....	173
2-66.	DEV Parity Control Register 1 (DEVCR1) Field Descriptions .....	174
2-67.	System Exception Control Register (SYSECR) Field Descriptions .....	175
2-68.	System Exception Status Register (SYSESR) Field Descriptions .....	176
2-69.	System Test Abort Status Register (SYSTASR) Field Descriptions .....	177
2-70.	Global Status Register (GLBSTAT) Field Descriptions .....	178
2-71.	Device Identification Register (DEVID) Field Descriptions .....	179
2-72.	Software Interrupt Vector Register (SSIVECTOR) Field Descriptions .....	180
2-73.	System Software Interrupt Flag Register (SSIF) Field Descriptions .....	181
2-74.	Secondary System Control Registers Summary .....	182
2-75.	PLL Control Register 3 (PLLCTL3) Field Descriptions .....	183
2-76.	CPU Logic BIST Clock Prescaler (STCLKDIV) Field Descriptions .....	184
2-77.	Clock 2 Control Register (CLK2CNTRL) Field Descriptions .....	185
2-78.	Peripheral Asynchronous Clock Configuration 1 Register (VCLKACON1) Field Descriptions .....	186
2-79.	<b>Clock Slip Register (CLKSLIP) Field Descriptions .....</b>	<b>188</b>
2-80.	<b>EFUSE Controller Control Register (EFC_CTLREG) Field Descriptions .....</b>	<b>189</b>
2-81.	Die Identification Register, Lower Word (DIEIDL_REG0) Field Descriptions .....	190
2-82.	Die Identification Register, Upper Word (DIEIDH_REG1) Field Descriptions .....	191
2-83.	Die Identification Register, Lower Word (DIEIDL_REG2) Field Descriptions .....	192
2-84.	Die Identification Register, Upper Word (DIEIDH_REG3) Field Descriptions .....	193
2-85.	Peripheral Central Resource Control Registers Summary .....	194
2-86.	Peripheral Memory Protection Set Register 0 (PMPROTSET0) Field Descriptions .....	195
2-87.	Peripheral Memory Protection Set Register 1 (PMPROTSET1) Field Descriptions .....	196
2-88.	Peripheral Memory Protection Clear Register 0 (PMPROTCLR0) Field Descriptions .....	197
2-89.	Peripheral Memory Protection Clear Register 1 (PMPROTCLR1) Field Descriptions .....	198
2-90.	Peripheral Protection Set Register 0 (PPROTSET0) Field Descriptions .....	199
2-91.	Peripheral Protection Set Register 1 (PPROTSET1) Field Descriptions .....	200
2-92.	Peripheral Protection Set Register 2 (PPROTSET2) Field Descriptions .....	201
2-93.	Peripheral Protection Set Register 3 (PPROTSET3) Field Descriptions .....	202
2-94.	Peripheral Protection Clear Register 0 (PPROTCLR0) Field Descriptions .....	203
2-95.	Peripheral Protection Clear Register 1 (PPROTCLR1) Field Descriptions .....	204
2-96.	Peripheral Protection Clear Register 2 (PPROTCLR2) Field Descriptions .....	205



2-97.	Peripheral Protection Clear Register 3 (PPROTCLR3) Field Descriptions .....	206
2-98.	Peripheral Memory Power-Down Set Register 0 (PCSPWRDWNSET0) Field Descriptions .....	207
2-99.	Peripheral Memory Power-Down Set Register 1 (PCSPWRDWNSET1) Field Descriptions .....	208
2-100.	Peripheral Memory Power-Down Clear Register 0 (PCSPWRDWNCLR0) Field Descriptions .....	209
2-101.	Peripheral Memory Power-Down Set Register 1 (PCSPWRDWNCLR1) Field Descriptions .....	210
2-102.	Peripheral Power-Down Set Register 0 (PSPWRDWNSET0) Field Descriptions .....	211
2-103.	Peripheral Power-Down Set Register 1 (PSPWRDWNSET1) Field Descriptions .....	212
2-104.	Peripheral Power-Down Set Register 2 (PSPWRDWNSET2) Field Descriptions .....	213
2-105.	Peripheral Power-Down Set Register 3 (PSPWRDWNSET3) Field Descriptions .....	214
2-106.	Peripheral Power-Down Clear Register 0 (PSPWRDWNCLR0) Field Descriptions .....	215
2-107.	Peripheral Power-Down Clear Register 1 (PSPWRDWNCLR1) Field Descriptions .....	216
2-108.	Peripheral Power-Down Clear Register 2 (PSPWRDWNCLR2) Field Descriptions .....	217
2-109.	Peripheral Power-Down Clear Register 3 (PSPWRDWNCLR3) Field Descriptions .....	218
3-1.	PMM Register Summary .....	226
3-2.	Logic Power Domain Control Register (LOGICPDPWRCTRL0) Field Descriptions .....	227
3-3.	Memory Power Domain Control Register 0 (MEMDPDPWRCTRL0) Field Descriptions .....	229
3-4.	Power Domain Clock Disable Register (PDCLKDISREG) Field Descriptions .....	230
3-5.	Power Domain Clock Disable Set Register (PDCLKDISSETREG) Field Descriptions .....	231
3-6.	Power Domain Clock Disable Clear Register (PDCLKDISCLRREG) Field Descriptions .....	232
3-7.	Logic Power Domain PD2 Power Status Register (LOGICPDPWRSTAT0) Field Descriptions .....	233
3-8.	Logic Power Domain PD3 Power Status Register (LOGICPDPWRSTAT1) Field Descriptions .....	234
3-9.	Logic Power Domain PD4 Power Status Register (LOGICPDPWRSTAT2) Field Descriptions .....	235
3-10.	Logic Power Domain PD5 Power Status Register (LOGICPDPWRSTAT3) Field Descriptions .....	236
3-11.	Memory Power Domain RAM_PD1 Power Status Register (MEMDPDPWRSTAT0) Field Descriptions .....	237
3-12.	Memory Power Domain RAM_PD2 Power Status Register (MEMDPDPWRSTAT1) Field Descriptions .....	238
3-13.	Memory Power Domain RAM_PD3 Power Status Register (MEMDPDPWRSTAT2) Field Descriptions .....	239
3-14.	Global Control Register 1 (GLOBALCTRL1) Field Descriptions .....	240
3-15.	Global Status Register (GLOBALSTAT) Field Descriptions .....	241
3-16.	PSCON Diagnostic Compare Key Register (PRCKEYREG) Field Descriptions .....	242
3-17.	LogicPD PSCON Diagnostic Compare Status Register 1 (LPDDCSTAT1) Field Descriptions .....	243
3-18.	LogicPD PSCON Diagnostic Compare Status Register 2 (LPDDCSTAT2) Field Descriptions .....	244
3-19.	Memory PD PSCON Diagnostic Compare Status Register 1 (MPDDCSTAT1) Field Descriptions .....	245
3-20.	Memory PD PSCON Diagnostic Compare Status Register 2 (MPDDCSTAT2) Field Descriptions .....	246
3-21.	Isolation Diagnostic Status Register (ISODIAGSTAT) Field Descriptions .....	247
4-1.	Input Multiplexing on 144QFP Parts .....	252
4-2.	IOMM Register Summary .....	255
4-3.	Revision Register Field Descriptions .....	256
4-4.	Boot Mode Register Field Descriptions .....	257
4-5.	Kicker Register 0 Field Descriptions .....	258
4-6.	Kicker Register 1 Field Descriptions .....	259
4-7.	Error Raw Status / Set Register Field Descriptions .....	260
4-8.	Error Signaling Enabled Status / Clear Register Field Descriptions .....	261
4-9.	Error Enable Register Field Descriptions .....	262
4-10.	Interrupt Enable Clear Register Field Descriptions .....	263
4-11.	Fault Address Register Field Descriptions .....	264
4-12.	Fault Status Register Field Descriptions .....	265
4-13.	FAULT_CLEAR_REG: Fault Clear Register Field Descriptions .....	266
4-14.	Pin Multiplexing Control Registers Field Descriptions .....	267
4-15.	Multiplexing and Control on Microcontrollers .....	268

5-1.	Flash Control Register Summary .....	277
5-2.	Flash Option Control Register (FRDCNTL) Field Descriptions .....	278
5-3.	Flash Error Detection and Correction Control Register 1 (FEDACCTRL1) Field Descriptions .....	279
5-4.	Flash Error Correction Control and Correction Register 2 (FEDACCTRL2) Field Descriptions .....	281
5-5.	Flash Un-correctable Error Address (FUNC_ERR_ADD) Field Descriptions .....	282
5-6.	Flash Bank Protection Register (FBPROT) Field Descriptions .....	283
5-7.	Flash Bank Sector Enable Register (FBSE) Field Descriptions .....	284
5-8.	Flash Bank Access Control Register (FBAC) Field Descriptions .....	285
5-9.	Flash Bank Fallback Power Register (FBFALLBACK) Field Descriptions.....	286
5-10.	Flash Pump Access Control Register 1 (FPAC1) Field Descriptions .....	287
5-11.	Flash Pump Access Control Register 2 (FPAC2) Field Descriptions .....	288
5-12.	Flash Module Access Control Register (FMAC) Field Descriptions .....	289
5-13.	Flash Module Status Register (FMSTAT) Field Descriptions .....	290
5-14.	Flash Module Access Control Register (FMAC) Field Descriptions .....	292
5-15.	EPROM Emulation Configuration Register (EEPROM_CONFIG) Field Descriptions .....	293
6-1.	TCRAM Module Control and Status Register Map .....	302
6-2.	TCRAM Module Control Register (RAMCTRL) Field Descriptions .....	303
6-3.	TCRAM Module Single-Bit Error Correction Threshold Register (RAMTHRESHOLD) Field Descriptions...	305
6-4.	TCRAM Module Single-Bit Error Occurrences Counter Register (RAMOCCUR) Field Descriptions .....	306
6-5.	TCRAM Module Interrupt Control Register (RAMINTCTRL) Field Descriptions.....	307
6-6.	TCRAM Module Error Status Register (RAMERRSTATUS) Field Descriptions .....	308
6-7.	TCRAM Module Single-Bit Error Address Register (RAMSERRADDR) Field Descriptions .....	309
6-8.	TCRAM Module Uncorrectable Error Address Register (RAMUERRADDR) Field Descriptions .....	310
6-9.	TCRAM Module Test Mode Control Register (RAMTEST) Field Descriptions .....	311
6-10.	TCRAM Module Test Mode Vector Register (RAMADDRDEVECT) Field Descriptions.....	312
6-11.	TCRAM Module Parity Error Address Register (RAMPERRADDR) Field Descriptions .....	313
7-1.	RAM Grouping.....	318
7-2.	Algorithm Mapping .....	318
7-3.	PBIST Registers .....	326
7-4.	RAM Configuration Register (RAMT) Field Descriptions .....	327
7-5.	Datalogger Register (DLR) Field Descriptions .....	328
7-6.	PBIST Activate/ROM Clock Enable Register (PACT) Field Descriptions .....	329
7-7.	PBIST ID Register Field Descriptions .....	330
7-8.	Override Register (OVER) Field Descriptions.....	331
7-9.	Fail Status Fail Register 0 (FSRF0) Field Descriptions .....	332
7-10.	Fail Status Fail Register 1 (FSRF1) Field Descriptions .....	332
7-11.	Fail Status Count 0 Register (FSRC0) Field Descriptions.....	333
7-12.	Fail Status Count Register 1 (FSRC1) Field Descriptions.....	333
7-13.	Fail Status Address 0 Register (FSRA0) Field Descriptions .....	334
7-14.	Fail Status Address 1 Register (FSRA1) Field Descriptions .....	334
7-15.	Fail Status Data Register 0 (FSRDL0) Field Descriptions.....	335
7-16.	Fail Status Data Register 1 (FSRDL1) Field Descriptions.....	335
7-17.	ROM Mask Register (ROM) Field Descriptions.....	336
7-18.	Algorithm Mask Register (ALGO) Field Descriptions.....	337
7-19.	RAM Info Mask Lower Register (RINFOL) Field Descriptions .....	338
7-20.	RAM Info Mask Upper Register (RINFOU) Field Descriptions.....	339
8-1.	STC Test Coverage and Duration .....	348
8-2.	Typical STC Execution Times .....	348
8-3.	STC Control Registers .....	349

8-4.	STC Global Control Register0 (STCGCR0) Field Descriptions .....	350
8-5.	STCGCR1 (STC Global Control Register1) Field Descriptions .....	351
8-6.	Self-Test Run Timeout Counter Preload Register (STCTPR).....	352
8-7.	STC Current ROM Address Register (STC_CADDR) Field Descriptions .....	353
8-8.	STC Current Interval Count Register (STCCICR) Field Descriptions .....	354
8-9.	Self-Test Global Status Register (STCGSTAT) Field Descriptions .....	355
8-10.	Self-Test Fail Status Register (STCFSTAT) Field Descriptions .....	356
8-11.	CPU1 Current MISR Register (CPU1_CURMISR[3:0]) Field Descriptions.....	357
8-12.	CPU2 Current MISR Register (CPU2_CURMISR[3:0]) Field Descriptions.....	359
9-1.	Compare Match Test Sequence .....	366
9-2.	Compare Mismatch Test Sequence .....	366
9-3.	Module Registers .....	368
9-4.	CCM-R4F Status Register (CCMSR) Field Descriptions .....	369
9-5.	CCM-R4F Key Register (CCMKEYR) Field Descriptions.....	370
10-1.	Valid Frequency Ranges for PLL .....	379
10-2.	PLL Value Encoding .....	379
10-3.	State Machine Timings .....	383
10-4.	PLL Module Registers.....	386
10-5.	LPOCLKDET Module Registers.....	386
10-6.	PLL Control 1 Register (PLLCTL1) Field Descriptions.....	387
10-7.	PLL Control 2 Register (PLLCTL2) Field Descriptions.....	389
10-8.	PLL Control 3 Register (PLLCTL3) Field Descriptions.....	391
10-9.	PLL Clock Slip Control Register (CLKSLIP) Field Descriptions .....	393
10-10.	General Purpose Register1 (GPREG1) Field Descriptions .....	394
10-11.	SSW PLL BIST Control Register 1 (SSWPLL1) Field Descriptions .....	395
10-12.	SSW PLL BIST Control Register 2 (SSWPLL2) Field Descriptions .....	397
10-13.	SSW PLL BIST Control Register 3 (SSWPLL3) Field Descriptions .....	398
10-14.	LPO/Clock Monitor Control Register (LPOMONCTL) Field Descriptions.....	399
10-15.	Clock Test Register (CLKTEST) Field Descriptions .....	402
11-1.	Clock Source Selection for DCC1 Counter0 .....	414
11-2.	Clock Source Selection for DCC1 Counter1 .....	414
11-3.	Clock Source Selection for DCC2 Counter0 .....	414
11-4.	Clock Source Selection for DCC2 Counter1 .....	414
11-5.	DCC Registers .....	415
11-6.	DCC Control Register (DCCCTRL) Field Descriptions .....	416
11-7.	DCC Revision Id Register (DCCREV) Field Descriptions .....	417
11-8.	DCC Counter0 Seed Register (DCCCNT0SEED) Field Descriptions .....	418
11-9.	DCC Valid0 Seed Register (DCCVALID0SEED) Field Descriptions .....	419
11-10.	DCC Counter1 Seed Register (DCCCNT0SEED) Field Descriptions .....	420
11-11.	DCC Status Register (DCCSTAT) Field Descriptions .....	421
11-12.	DCC Counter0 Seed Register (DCCCNT0SEED) Field Descriptions .....	422
11-13.	DCC Valid0 Value Register (DCCVALID0) Field Descriptions .....	423
11-14.	DCC Counter1 Value Register (DCCCNT1) Field Descriptions .....	424
11-15.	DCC Counter1 Value Register (DCCCNT1) Field Descriptions .....	425
11-16.	DCC Counter1 Value Register (DCCCNT1) Field Descriptions .....	426
12-1.	ESM interrupt and <b>ERROR</b> pin behavior .....	428
12-2.	ESM Module Registers.....	434
12-3.	ESM Enable <b>ERROR</b> Pin Action/Response Register 1 (ESMEEPAPR1) Field Descriptions.....	435
12-4.	ESM Disable <b>ERROR</b> Pin Action/Response Register 1 (ESMDEPAPR1) Field Descriptions .....	436

12-5.	ESM Interrupt Enable Set/Status Register 1 (ESMIESR1) Field Descriptions.....	437
12-6.	ESM Interrupt Enable Clear/Status Register 1 (ESMIECR1) Field Descriptions .....	438
12-7.	ESM Interrupt Level Set/Status Register 1 (ESMILSR1) Field Descriptions .....	439
12-8.	ESM Interrupt Level Clear/Status Register 1 (ESMILCR1) Field Descriptions .....	440
12-9.	ESM Status Register 1 (ESMSR1) Field Descriptions .....	441
12-10.	ESM Status Register 2 (ESMSR2) Field Descriptions .....	442
12-11.	ESM Status Register 3 (ESMSR3) Field Descriptions .....	443
12-12.	ESM <del>ERROR</del> Pin Status Register (ESMEPSR) Field Descriptions .....	444
12-13.	ESM Interrupt Offset High Register (ESMIOFFHR) Field Descriptions.....	445
12-14.	ESM Interrupt Offset Low Register (ESMIOFFLR) Field Descriptions.....	446
12-15.	ESM Low-Time Counter Register (ESMLTCR) Field Descriptions .....	447
12-16.	ESM Low-Time Counter Preload Register (ESMLTCPR) Field Descriptions .....	448
12-17.	ESM Error Key Register (ESMEKR) Field Descriptions .....	449
12-18.	ESM Status Shadow Register 2 (ESMSSR2) Field Descriptions.....	450
12-19.	ESM Influence <del>ERROR</del> Pin Set/Status Register 4 (ESMIEPSR4) Field Descriptions .....	451
12-20.	ESM Influence <del>ERROR</del> Pin Clear/Status Register 4 (ESMIEPSR4) Field Descriptions .....	452
12-21.	ESM Interrupt Enable Set/Status Register 4 (ESMIESR4) Field Descriptions.....	453
12-22.	ESM Interrupt Enable Clear/Status Register 4 (ESMIECR4) Field Descriptions .....	454
12-23.	ESM Interrupt Level Set/Status Register 4 (ESMILSR4) Field Descriptions .....	455
12-24.	ESM Interrupt Level Clear/Status Register 4 (ESMILCR4) Field Descriptions .....	456
12-25.	ESM Status Register 4 (ESMSR4) Field Descriptions .....	457
13-1.	RTI Registers .....	471
13-2.	RTI Global Control Register (RTIGCTRL) Field Descriptions.....	473
13-3.	RTI Timebase Control Register (RTITBCTRL) Field Descriptions .....	474
13-4.	RTI Capture Control Register (RTICAPCTRL) Field Descriptions .....	475
13-5.	RTI Compare Control Register (RTICOMPCTRL) Field Descriptions .....	476
13-6.	RTI Free Running Counter 0 Register (RTIFRC0) Field Descriptions .....	477
13-7.	RTI Up Counter 0 Register (RTIUC0) Field Descriptions.....	478
13-8.	RTI Compare Up Counter 0 Register (RTICPUC0) Field Descriptions .....	479
13-9.	RTI Capture Free Running Counter 0 Register (RTICAFRC0) Field Descriptions .....	480
13-10.	RTI Capture Up Counter 0 Register (RTICAUC0) Field Descriptions .....	481
13-11.	RTI Free Running Counter 1 Register (RTIFRC1) Field Descriptions .....	482
13-12.	RTI Up Counter 1 Register (RTIUC1) Field Descriptions.....	483
13-13.	RTI Compare Up Counter 1 Register (RTICPUC1) Field Descriptions .....	484
13-14.	RTI Capture Free Running Counter 1 Register (RTICAFRC1) Field Descriptions .....	485
13-15.	RTI Capture Up Counter 1 Register (RTICAUC1) Field Descriptions .....	486
13-16.	RTI Compare 0 Register (RTICOMP0) Field Descriptions .....	487
13-17.	RTI Update Compare 0 Register (RTIUDCP0) Field Descriptions .....	488
13-18.	RTI Compare 1 Register (RTICOMP1) Field Descriptions .....	489
13-19.	RTI Update Compare 1 Register (RTIUDCP1) Field Descriptions .....	490
13-20.	RTI Compare 2 Register (RTICOMP2) Field Descriptions .....	491
13-21.	RTI Update Compare 2 Register (RTIUDCP2) Field Descriptions .....	492
13-22.	RTI Compare 3 Register (RTICOMP3) Field Descriptions .....	493
13-23.	RTI Update Compare 3 Register (RTIUDCP3) Field Descriptions .....	494
13-24.	RTI Timebase Low Compare Register (RTITBLCOMP) Field Descriptions.....	495
13-25.	RTI Timebase High Compare Register (RTITBHCMP) Field Descriptions.....	496
13-26.	RTI Set Interrupt Control Register (RTISETINTENA) Field Descriptions.....	497
13-27.	RTI Clear Interrupt Control Register (RTICLEARINTENA) Field Descriptions .....	499
13-28.	RTI Interrupt Flag Register (RTIINTFLAG) Field Descriptions.....	501

13-29. Digital Watchdog Control Register (RTIDWDCTRL) Field Descriptions .....	503
13-30. Digital Watchdog Preload Register (RTIDWDPRLD) Field Descriptions .....	504
13-31. Watchdog Status Register (RTIWDSTATUS) Field Descriptions .....	505
13-32. RTI Watchdog Key Register (RTIDWDKEY) Field Descriptions .....	506
13-33. Example of a WDKEY Sequence .....	506
13-34. RTI Watchdog Down Counter Register (RTIDWDCNTR) Field Descriptions .....	507
13-35. Digital Windowed Watchdog Reaction Control (RTIWWDRXNCTRL) Field Descriptions.....	508
13-36. Digital Windowed Watchdog Window Size Control (RTIWWDSIZECTRL) Field Descriptions .....	509
14-1. CRC Modes in which DMA Request and Counter Logic are Active or Inactive .....	520
14-2. Modes in which Interrupt Condition Can Occur .....	521
14-3. Interrupt Offset Mapping .....	523
14-4. CRC Control Registers .....	529
14-5. CRC Global Control Register 0 (CRC_CTRL0) Field Descriptions.....	531
14-6. CRC Global Control Register 1 (CRC_CTRL1) Field Descriptions.....	532
14-7. CRC Global Control Register 2 (CRC_CTRL2) Field Descriptions.....	533
14-8. CRC Interrupt Enable Set Register (CRC_INTS) Field Descriptions .....	535
14-9. CRC Interrupt Enable Reset Register (CRC_INTR) Field Descriptions .....	540
14-10. CRC Interrupt Status Register (CRC_STATUS) Field Descriptions.....	545
14-11. CRC Interrupt Offset (CRC_INT_OFFSET_REG) Field Descriptions.....	548
14-12. CRC Busy Register (CRC_BUSY) Field Descriptions .....	549
14-13. CRC Pattern Counter Preload Register 1 (CRC_PCOUNT_REG1) Field Descriptions .....	550
14-14. CRC Sector Counter Preload Register 1 (CRC_SCOUNT_REG1) Field Descriptions .....	551
14-15. CRC Current Sector Register 1 (CRC_CURSEC_REG1) Field Descriptions.....	552
14-16. CRC Channel 1 Watchdog Timeout Preload Register A (CRC_WDTPLD1) Field Descriptions .....	553
14-17. CRC Channel 1 Block Complete Timeout Preload Register B (CRC_BCTOPLD1) Field Descriptions .....	554
14-18. Channel 1 PSA Signature Low Register (PSA_SIGREGL1) Field Descriptions.....	555
14-19. Channel 1 PSA Signature High Register (PSA_SIGREGH1) Field Descriptions.....	556
14-20. Channel 1 CRC Value Low Register (CRC_REGL1) Field Descriptions .....	557
14-21. Channel 1 CRC Value High Register (CRC_REGH1) Field Descriptions .....	558
14-22. Channel 1 PSA Sector Signature Low Register (PSA_SECSIGREGL1) Field Descriptions .....	559
14-23. Channel 1 PSA Sector Signature High Register (PSA_SECSIGREGH1) Field Descriptions .....	560
14-24. Channel 1 Raw Data Low Register (RAW_DATAREGL1) Field Descriptions .....	561
14-25. Channel 1 Raw Data High Register (RAW_DATAREGH1) Field Descriptions .....	562
14-26. CRC Pattern Counter Preload Register 2 (CRC_PCOUNT_REG2) Field Descriptions .....	563
14-27. CRC Sector Counter Preload Register 2 (CRC_SCOUNT_REG2) Field Descriptions .....	564
14-28. CRC Current Sector Register 2 (CRC_CURSEC_REG2) Field Descriptions.....	565
14-29. CRC Channel 2 Watchdog Timeout Preload Register A (CRC_WDTPLD2) Field Descriptions .....	566
14-30. CRC Channel 2 Block Complete Timeout Preload Register B (CRC_BCTOPLD2) Field Descriptions .....	567
14-31. Channel 2 PSA Signature Low Register (PSA_SIGREGL2) Field Descriptions.....	568
14-32. Channel 2 PSA Signature High Register (PSA_SIGREGH2) Field Descriptions.....	569
14-33. Channel 2 CRC Value Low Register (CRC_REGL2) Field Descriptions .....	570
14-34. Channel 2 CRC Value High Register (CRC_REGH2) Field Descriptions .....	571
14-35. Channel 2 PSA Sector Signature Low Register (PSA_SECSIGREGL2) Field Descriptions .....	572
14-36. Channel 2 PSA Sector Signature High Register (PSA_SECSIGREGH2) Field Descriptions .....	573
14-37. Channel 2 Raw Data Low Register (RAW_DATAREGL2) Field Descriptions .....	574
14-38. Channel 2 Raw Data High Register (RAW_DATAREGH2) Field Descriptions .....	575
14-39. CRC Pattern Counter Preload Register 3 (CRC_PCOUNT_REG3) Field Descriptions .....	576
14-40. CRC Sector Counter Preload Register 3 (CRC_SCOUNT_REG3) Field Descriptions.....	577
14-41. CRC Current Sector Register 3 (CRC_CURSEC_REG3) Field Descriptions.....	578



14-42. CRC Channel 3 Watchdog Timeout Preload Register A (CRC_WDTPLD3) Field Descriptions .....	579
14-43. CRC Channel 3 Block Complete Timeout Preload Register B (CRC_BCTOPLD3) Field Descriptions .....	580
14-44. Channel 3 PSA Signature Low Register (PSA_SIGREGL3) Field Descriptions.....	581
14-45. Channel 3 PSA Signature High Register (PSA_SIGREGH3) Field Descriptions.....	582
14-46. Channel 3 CRC Value Low Register (CRC_REGL3) Field Descriptions .....	583
14-47. Channel 3 CRC Value High Register (CRC_REGH3) Field Descriptions .....	584
14-48. Channel 3 PSA Sector Signature Low Register (PSA_SECSIGREGL3) Field Descriptions .....	585
14-49. Channel 3 PSA Sector Signature High Register (PSA_SECSIGREGH3) Field Descriptions .....	586
14-50. Channel 3 Raw Data Low Register (RAW_DATAREGL3) Field Descriptions .....	587
14-51. Channel 3 Raw Data High Register (RAW_DATAREGH3) Field Descriptions .....	588
14-52. CRC Pattern Counter Preload Register 4 (CRC_PCOUNT_REG4) Field Descriptions .....	589
14-53. CRC Sector Counter Preload Register 4 (CRC_SCOUNT_REG4) Field Descriptions .....	590
14-54. CRC Current Sector Register 4 (CRC_CURSEC_REG4) Field Descriptions.....	591
14-55. CRC Channel 4 Watchdog Timeout Preload Register A (CRC_WDTPLD4) Field Descriptions .....	592
14-56. CRC Channel 4 Block Complete Timeout Preload Register B (CRC_BCTOPLD4) Field Descriptions .....	593
14-57. Channel 4 PSA Signature Low Register (PSA_SIGREGL4) Field Descriptions.....	594
14-58. Channel 4 PSA Signature High Register (PSA_SIGREGH4) Field Descriptions.....	595
14-59. Channel 4 CRC Value Low Register (CRC_REGL4) Field Descriptions .....	596
14-60. Channel 4 CRC Value High Register (CRC_REGH4) Field Descriptions .....	597
14-61. Channel 4 PSA Sector Signature Low Register (PSA_SECSIGREGL4) Field Descriptions .....	598
14-62. Channel 4 PSA Sector Signature High Register (PSA_SECSIGREGH4) Field Descriptions .....	599
14-63. Channel 4 Raw Data Low Register (RAW_DATAREGL4) Field Descriptions .....	600
14-64. Channel 4 Raw Data High Register (RAW_DATAREGH4) Field Descriptions .....	601
14-65. Channel 4 Data Bus Selection Register Field Descriptions .....	602
15-1. VIM Control Register Summary .....	619
15-2. Interrupt Vector Table Parity Flag Register (PARFLG) Field Descriptions .....	620
15-3. Interrupt Vector Table Parity Control Register (PARCTL) Field Descriptions .....	621
15-4. Address Parity Error Register (PARERR) Field Descriptions .....	622
15-5. Fall Back Address Parity Error Register (FBPARERR) Field Descriptions .....	623
15-6. Interrupt Dispatch .....	624
15-7. IRQ Index Offset Vector Register (IRQINDEX) Field Descriptions .....	625
15-8. FIQ Index Offset Vector Register (FIQINDEX) Field Descriptions .....	626
15-9. FIQ/IRQ Program Control Registers[0] (FIRQPR[0]) Field Descriptions .....	627
15-10. Pending Interrupt Read Registers[0] (INTREQ[0]) Field Descriptions .....	628
15-11. Interrupt Enable Set Register[0] (REQENASET[0]) Field Descriptions .....	629
15-12. Request Enable Clear Register (REQENACLR) Field Descriptions .....	630
15-13. Wake-Up Enable Set Registers[0] (WAKEENASET[0]) Field Descriptions .....	631
15-14. Wake-Up Enable Clear Registers[0] (WAKEENACLR[0]) Field Descriptions.....	632
15-15. IRQ Interrupt Vector Register (IRQVECREG) Field Descriptions .....	633
15-16. FIQ Interrupt Vector Register (FIQVECREG) Field Descriptions .....	634
15-17. Capture Event Register (CAPEVT) Field Descriptions .....	635
15-18. Interrupt Control Registers Organization .....	636
15-19. Interrupt Control Registers[0:31] (CHANCTL[0:15] Field Descriptions .....	636
16-1. Arbitration According to Priority Queues and Priority Schemes .....	647
16-2. Maximum Number of DMA Transactions per Channel in Non-Bypass Mode.....	654
16-3. Maximum Number of DMA Transactions per Channel in Bypass Mode.....	654
16-4. Control Packet RAM.....	657
16-5. Control Packet RAM .....	657
16-6. Parity RAM.....	657



16-7. DMA Control Registers .....	658
16-8. Control Packet Memory Map .....	660
16-9. Global Control Register (GCTRL) Field Descriptions .....	661
16-10. Channel Pending Register (PEND) Field Descriptions .....	662
16-11. DMA Status Register (DMASTAT) Field Descriptions .....	663
16-12. HW Channel Enable Set and Status Register (HWCHENAS) Field Descriptions .....	664
16-13. HW Channel Enable Reset and Status Register (HWCHENAR) Field Descriptions.....	665
16-14. SW Channel Enable Set and Status Register (SWCHENAS) Field Descriptions .....	666
16-15. SW Channel Enable Reset and Status Register (SWCHENAR) Field Descriptions .....	667
16-16. Channel Priority Set Register (CHPRIOS) Field Descriptions .....	668
16-17. Channel Priority Reset Register (CHPRIOR) Field Descriptions .....	669
16-18. Global Channel Interrupt Enable Set Register (GCHIENAS) Field Descriptions .....	670
16-19. Global Channel Interrupt Enable Reset Register (GCHIENAR) Field Descriptions .....	671
16-20. DMA Request Assignment Register 0 (DREQASIO) Field Descriptions .....	672
16-21. DMA Request Assignment Register 1 (DREQASI1) Field Descriptions .....	673
16-22. DMA Request Assignment Register 2 (DREQASI2) Field Descriptions .....	674
16-23. DMA Request Assignment Register 3 (DREQASI3) Field Descriptions .....	675
16-24. Port Assignment Register 0 (PAR0) Field Descriptions .....	676
16-25. Port Assignment Register 1 (PAR1) Field Descriptions .....	678
16-26. FTC Interrupt Mapping Register (FTCMAP) Field Descriptions .....	680
16-27. LFS Interrupt Mapping Register (LFSMAP) Field Descriptions .....	681
16-28. HBC Interrupt Mapping Register (HBCMAP) Field Descriptions .....	682
16-29. BTC Interrupt Mapping Register (BTCMAP) Field Descriptions .....	683
16-30. BER Interrupt Mapping Register (BERMAP) Field Descriptions.....	684
16-31. FTC Interrupt Enable Set (FTCINTENAS) Field Descriptions .....	685
16-32. FTC Interrupt Enable Reset (FTCINTENAR) Field Descriptions .....	686
16-33. LFS Interrupt Enable Set (LFSINTENAS) Field Descriptions .....	687
16-34. LFS Interrupt Enable Reset (LFSINTENAR) Field Descriptions .....	688
16-35. HBC Interrupt Enable Set (HBCINTENAS) Field Descriptions.....	689
16-36. HBC Interrupt Enable Reset (HBCINTENAR) Field Descriptions .....	690
16-37. BTC Interrupt Enable Reset (BTCINTENAS) Field Descriptions .....	691
16-38. BTC Interrupt Enable Reset (BTCINTENAR) Field Descriptions .....	692
16-39. Global Interrupt Flag Register (GINTFLAG) Field Descriptions .....	693
16-40. FTC INTERRUPT FLAG Register (FTCFLAG) Field Descriptions .....	694
16-41. LFS Interrupt Flag Register (LFSFLAG) Field Descriptions .....	695
16-42. HBC Interrupt Flag (HBCFLAG) Field Descriptions .....	696
16-43. BTC Interrupt Flag Register (BTCFLAG) Field Descriptions .....	697
16-44. BER Interrupt Flag Register (BERFLAG) Field Descriptions.....	698
16-45. FTCA Interrupt Channel Offset Register (FTCAOFFSET) Field Descriptions .....	699
16-46. LFSA Interrupt Channel Offset Register (LFSAOFFSET) Field Descriptions .....	700
16-47. HBCA Interrupt Channel Offset Register (HBCAOFFSET) Field Descriptions .....	701
16-48. BTCA Interrupt Channel Offset Register (BTCAOFFSET) Field Descriptions .....	702
16-49. BERA Interrupt Channel Offset Register (BERAOFFSET) Field Descriptions.....	703
16-50. FTCB Interrupt Channel Offset Register (FTCBOFFSET) Field Descriptions .....	704
16-51. LFSB Interrupt Channel Offset Register (LFSBOFFSET) Field Descriptions .....	705
16-52. HBCB Interrupt Channel Offset Register (HBCBOFFSET) Field Descriptions .....	706
16-53. BTCB Interrupt Channel Offset Register (BTCBOFFSET) Field Descriptions .....	707
16-54. BERB Interrupt Channel Offset Register (BERBOFFSET) Field Descriptions.....	708
16-55. Port Control Register (PTCRL) Field Descriptions .....	709

16-56. RAM Test Control (RTCTRL) Field Descriptions .....	710
16-57. Debug Control (DCTRL) Field Descriptions .....	711
16-58. Watch Point Register (WPR) Field Descriptions .....	712
16-59. Watch Mask Register (WMR) Field Descriptions .....	713
16-60. Port A Active Channel Source Address Register (PAACSADDR) Field Descriptions .....	714
16-61. Port A Active Channel Destination Address Register (PAACDADDR) Field Descriptions .....	715
16-62. Port A Active Channel Transfer Count Register (PAACTC) Field Descriptions .....	716
16-63. Port B Active Channel Source Address Register (PBACDADDR) Field Descriptions .....	717
16-64. Port B Active Channel Destination Address Register (PBACDADDR) Field Descriptions .....	718
16-65. Port B Active Channel Transfer Count Register (PBACTC) Field Descriptions .....	719
16-66. Parity Control Register (DMAPCR) Field Descriptions.....	720
16-67. DMA Parity Error Address Register (DMAPAR) Field Descriptions .....	721
16-68. DMA Memory Protection Control Register.....	722
16-69. DMA Memory Protection Status Register (DMAMPST) Field Descriptions .....	724
16-70. DMA Memory Protection Region 0 Start Address Register (DMAMPR0S) Field Descriptions .....	725
16-71. DMA Memory Protection Region 0 End Address Register (DMAMPR0E) Field Descriptions .....	726
16-72. DMA Memory Protection Region 1 Start Address Register (DMAMPR1S) Field Descriptions .....	727
16-73. DMA Memory Protection Region 1 End Address Register (DMAMPR1E) Field Descriptions .....	728
16-74. DMA Memory Protection Region 2 Start Address Register (DMAMPR2S) Field Descriptions .....	729
16-75. DMA Memory Protection Region 2 End Address Register (DMAMPR2E) Field Descriptions .....	730
16-76. DMA Memory Protection Region 3 Start Address Register (DMAMPR3S) Field Descriptions .....	731
16-77. DMA Memory Protection Region 3 End Address Register (DMAMPR3E) Field Descriptions .....	732
16-78. Initial Source Address (ISADDR) Field Descriptions .....	733
16-79. Initial Destination Address Register (IDADDR) Field Descriptions .....	734
16-80. Initial Transfer Count Register (ITCOUNT) Field Descriptions.....	735
16-81. CHANNEL CONTROL Register (CHCTRL) Field Descriptions .....	736
16-82. Element Index Offset Register (EIOFF) Field Descriptions .....	737
16-83. Frame Index Offset Register (FIOFF) Field Descriptions.....	738
16-84. Current Source Address Register (CSADDR) Field Descriptions .....	739
16-85. Current Destination Address Register (CDADDR) Field Descriptions .....	740
16-86. Current Transfer Count Register (CTCOUNT) Field Descriptions.....	741
17-1. EMIF Pins Used to Access Both SDRAM and Asynchronous Memories.....	746
17-2. EMIF Pins Specific to SDRAM .....	746
17-3. EMIF Pins Specific to Asynchronous Memory .....	747
17-4. EMIF SDRAM Commands.....	748
17-5. Truth Table for SDRAM Commands .....	748
17-6. 16-bit EMIF Address Pin Connections .....	750
17-7. Description of the SDRAM Configuration Register (SDCR).....	751
17-8. Description of the SDRAM Refresh Control Register (SDRCR) .....	751
17-9. Description of the SDRAM Timing Register (SDTIMR) .....	752
17-10. Description of the SDRAM Self Refresh Exit Timing Register (SDSRETR).....	752
17-11. SDRAM LOAD MODE REGISTER Command.....	753
17-12. Refresh Urgency Levels.....	754
17-13. Mapping from Logical Address to EMIF Pins for 16-bit SDRAM .....	759
17-14. Normal Mode vs. Select Strobe Mode .....	760
17-15. Description of the Asynchronous <i>m</i> Configuration Register (CE <sub>n</sub> CFG) .....	762
17-16. Description of the Asynchronous Wait Cycle Configuration Register (AWCC) .....	763
17-17. Description of the EMIF Interrupt Mask Set Register (INTMSKSET) .....	764
17-18. Description of the EMIF Interrupt Mast Clear Register (INTMSKCLR) .....	764

17-19. Asynchronous Read Operation in Normal Mode .....	764
17-20. Asynchronous Write Operation in Normal Mode.....	766
17-21. Asynchronous Read Operation in Select Strobe Mode .....	768
17-22. Asynchronous Write Operation in Select Strobe Mode .....	770
17-23. Interrupt Monitor and Control Bit Fields .....	773
17-24. External Memory Interface (EMIF) Registers .....	778
17-25. Module ID Register (MIDR) Field Descriptions .....	779
17-26. Asynchronous Wait Cycle Configuration Register (AWCCR) Field Descriptions .....	780
17-27. SDRAM Configuration Register (SDCR) Field Descriptions .....	781
17-28. SDRAM Refresh Control Register (SDRCR) Field Descriptions.....	783
17-29. Asynchronous <i>n</i> Configuration Register (CE <sub><i>n</i></sub> CFG) Field Descriptions .....	784
17-30. SDRAM Timing Register (SDTIMR) Field Descriptions.....	785
17-31. SDRAM Self Refresh Exit Timing Register (SDSRETR) Field Descriptions .....	786
17-32. EMIF Interrupt Raw Register (INTRAW) Field Descriptions.....	787
17-33. EMIF Interrupt Mask Register (INTMSK) Field Descriptions .....	788
17-34. EMIF Interrupt Mask Set Register (INTMSKSET) Field Descriptions .....	789
17-35. EMIF Interrupt Mask Clear Register (INTMSKCLR) Field Descriptions .....	790
17-36. Page Mode Control Register (PMCR) Field Descriptions .....	791
17-37. SR Field Value For the EMIF to K4S641632H-TC(L)70 Interface.....	792
17-38. SDTIMR Field Calculations for the EMIF to K4S641632H-TC(L)70 Interface .....	794
17-39. RR Calculation for the EMIF to K4S641632H-TC(L)70 Interface.....	795
17-40. RR Calculation for the EMIF to K4S641632H-TC(L)70 Interface.....	795
17-41. SDCR Field Values For the EMIF to K4S641632H-TC(L)70 Interface .....	796
17-42. AC Characteristics for a Read Access .....	797
17-43. AC Characteristics for a Write Access .....	797
18-1. ADC Groups' Operating Mode Control and Status Registers .....	813
18-2. Calibration Reference Voltages <sup>†</sup> .....	821
18-3. Self-Test Reference Voltages <sup>†</sup> .....	824
18-4. Determination of ADC Input Channel Condition .....	825
18-5. Output Buffer, and Pull Control Behavior for ADxEVT as GPIO Pins .....	830
18-6. ADC Registers Summary .....	831
18-7. ADC Reset Control Register (ADRSTCR) Field Descriptions.....	833
18-8. ADC Operating Mode Control Register (ADOPMODECR) Field Descriptions .....	834
18-9. ADC Clock Control Register (ADCLOCKCR) Field Descriptions .....	836
18-10. ADC Calibration Mode Control Register (ADCALCR) Field Descriptions .....	837
18-11. ADC Event Group Operating Mode Control Register (ADEVMODECR) Field Descriptions .....	838
18-12. ADC Group1 Operating Mode Control Register (ADG1MODECR) Field Descriptions .....	840
18-13. ADC Group 2 Operating Mode Control Register (ADG2MODECR) Field Descriptions .....	842
18-14. ADC Event Group Trigger Source Select Register (ADEVSR) Field Descriptions .....	844
18-15. ADC Group1 Trigger Source Select Register (ADG1SRC) Field Descriptions .....	845
18-16. ADC Group2 Trigger Source Select Register (ADG2SRC) Field Descriptions .....	846
18-17. ADC Event Group Interrupt Enable Control Register (ADEVINTENA) Field Descriptions .....	847
18-18. ADC Group1 Interrupt Enable Control Register (ADG1INTENA) Field Descriptions .....	848
18-19. ADC Group2 Interrupt Enable Control Register (ADG2INTENA) Field Descriptions .....	849
18-20. ADC Event Group Interrupt Flag Register (ADEVINTFLG) Field Descriptions .....	850
18-21. ADC Group1 Interrupt Flag Register (ADG1INTFLG) Field Descriptions .....	851
18-22. ADC Group2 Interrupt Flag Register (ADG2INTFLG) Field Descriptions .....	852
18-23. ADC Event Group Threshold Interrupt Control Register (ADEVTHRINTCR) Field Descriptions.....	853
18-24. ADC Group1 Threshold Interrupt Control Register (ADG1THRINTCR) Field Descriptions .....	854

18-25. ADC Group2 Threshold Interrupt Control Register (ADG2THRINTCR) Field Descriptions .....	855
18-26. ADC Event Group DMA Control Register (ADEVDMACR) Field Descriptions .....	856
18-27. ADC Group1 DMA Control Register (ADG1DMACR) Field Descriptions.....	857
18-28. ADC Group2 DMA Control Register (ADG2DMACR) Field Descriptions.....	858
18-29. ADC Results Memory Configuration Register (ADBNDCCR) Field Descriptions .....	859
18-30. ADC Results Memory Size Configuration Register (ADBNDEND) Field Descriptions.....	860
18-31. ADC Event Group Sampling Time Configuration Register (ADEVSAMP) Field Descriptions .....	861
18-32. ADC Group1 Sampling Time Configuration Register (ADG1SAMP) Field Descriptions .....	862
18-33. ADC Group2 Sampling Time Configuration Register (ADG2SAMP) Field Descriptions .....	863
18-34. ADC Event Group Status Register (ADEVSR) Field Descriptions .....	864
18-35. ADC Group1 Status Register (ADG1SR) Field Descriptions.....	865
18-36. ADC Group2 Status Register (ADG2SR) Field Descriptions.....	866
18-37. ADC Event Group Channel Select Register (ADEVSEL) Field Descriptions .....	867
18-38. ADC Group1 Channel Select Register (ADG1SEL) Field Descriptions.....	868
18-39. ADC Group2 Channel Select Register (ADG2SEL) Field Descriptions.....	869
18-40. ADC Calibration and Error Offset Correction Register (ADCALR) Field Descriptions .....	870
18-41. ADC State Machine Status Register (ADSMSTATE) Field Descriptions .....	871
18-42. ADC Channel Last Conversion Value Register (ADLASTCONV) Field Descriptions.....	872
18-43. ADC Event Group Results' FIFO Register (ADEVBUFFER) Field Descriptions.....	873
18-44. ADC Group1 Results FIFO Register (ADG1BUFFER) Field Descriptions .....	874
18-45. ADC Group2 Results FIFO Register (ADG2BUFFER) Field Descriptions .....	875
18-46. ADC ADEVT Pin Direction Control Register (ADEVTDIR) Field Descriptions .....	877
18-47. ADC ADEVT Pin Output Value Control Register (ADEVTOUT) Field Descriptions.....	878
18-48. ADC ADEVT Pin Input Value Register (ADEVTIN) Field Descriptions .....	879
18-49. ADC ADEVT Pin Set Register (ADEVTSET) Field Descriptions .....	880
18-50. ADC ADEVT Pin Clear Register (ADEVTCLR) Field Descriptions .....	881
18-51. ADC ADEVT Pin Open Drain Enable Register (ADEVTPDR) Field Descriptions .....	882
18-52. ADC ADEVT Pin Pull Control Disable Register (ADEVTPDIS) Field Descriptions.....	883
18-53. ADC ADEVT Pin Pull Control Select Register (ADEVTPSEL) Field Descriptions .....	884
18-54. ADC Event Group Sample Cap Discharge Control Register (ADEVSAMPDISEN) Field Descriptions.....	885
18-55. ADC Group1 Sample Cap Discharge Control Register (ADG1SAMPDISEN) Field Descriptions.....	886
18-56. ADC Group2 Sample Cap Discharge Control Register (ADG2SAMPDISEN) Field Descriptions.....	887
18-57. ADC Magnitude Compare Interrupt Control Registers (ADMAGINTxCR) Field Descriptions .....	888
18-58. ADC Magnitude Compare Mask Register (ADMAGINTxMASK) Field Descriptions .....	889
18-59. ADC Magnitude Compare Interrupt Enable Set Register (ADMAGINTENASET) Field Descriptions.....	890
18-60. ADC Magnitude Compare Interrupt Enable Clear Register (ADMAGINTENACLAR) Field Descriptions .....	891
18-61. ADC Magnitude Compare Interrupt Flag Register (ADMAGINTFLG) Field Descriptions .....	892
18-62. ADC Magnitude Compare Interrupt Offset Register (ADMAGINTOFF) Field Descriptions .....	893
18-63. ADC Event Group FIFO Reset Control Register (ADEVFIFORESETCR) Field Descriptions .....	894
18-64. ADC Group1 FIFO Reset Control Register (ADG1FIFORESETCR) Field Descriptions .....	895
18-65. ADC Group2 FIFO Reset Control Register (ADG2FIFORESETCR) Field Descriptions .....	896
18-66. ADC Event Group RAM Write Address Register (ADEVRAMWRADDR) Field Descriptions .....	897
18-67. ADC Group1 RAM Write Address Register (ADG1RAMWRADDR) Field Descriptions .....	898
18-68. ADC Group2 RAM Write Address Register (ADG2RAMWRADDR) Field Descriptions .....	899
18-69. ADC Parity Control Register (ADPARCR) Field Descriptions.....	900
18-70. ADC Parity Error Address Register (ADPARADDR) Field Descriptions.....	901
18-71. ADC Power-Up Delay Control Register (ADPWRUPDLYCTRL) Field Descriptions .....	902
19-1. N2HET RAM Base Addresses .....	912
19-2. N2HET RAM Bank Structure .....	913

19-3. Pin Safe State Upon Parity Error Detection .....	914
19-4. N2HET Parity Bit Mapping .....	914
19-5. Prescale Factor Register Encoding.....	916
19-6. Interpretation of the 7-Bit HR Data Field .....	917
19-7. Edge Detection Input Timing for Loop Resolution Instructions .....	925
19-8. Edge Detection Input Timing for High Resolution Instructions.....	926
19-9. Input Buffer, Output Buffer and Pull Control Behavior .....	931
19-10. N2HET Pin Disable Feature .....	932
19-11. Pulse Length Examples for Suppression Filter .....	933
19-12. Interrupt Sources and Corresponding Offset Values in Registers HETOFFx .....	933
19-13. N2HET Registers .....	941
19-14. Global Configuration Register (HETGCR) .....	942
19-15. Prescale Factor Register (HETPFR) .....	944
19-16. N2HET Current Address (HETADDR) .....	945
19-17. Offset Index Priority Level 1 Register (HETOFF1) .....	946
19-18. Interrupt Offset Encoding Format .....	946
19-19. Offset Index Priority Level 2 Register (HETOFF2) .....	947
19-20. Interrupt Enable Set Register (HETINTENAS).....	948
19-21. NHET Interrupt Enable Clear (HETINTENAC) .....	949
19-22. Exception Control Register 1 (HETEXC1) .....	950
19-23. Exception Control Register 2 (HETEXC2) .....	951
19-24. Interrupt Priority Register (HETPRY).....	952
19-25. Interrupt Flag Register (HETFLG) .....	953
19-26. AND Share Control Register (HETAND).....	954
19-27. HR Share Control Register (HETHRSH).....	955
19-28. XOR Share Control Register (HETXOR) .....	956
19-29. Request Enable Set Register (HETREQENS) .....	957
19-30. Request Enable Clear Register (HETREQENC).....	958
19-31. Request Destination Select Register (HETREQDS) .....	959
19-32. N2HET Direction Register (HETDIR).....	960
19-33. N2HET Data Input Register (HETDIN).....	961
19-34. N2HET Data Output Register (R-Write) (HETDOUT) .....	962
19-35. N2HET Data Set Register (HETDSET) .....	963
19-36. N2HET Data Clear Register (HETDCLR).....	964
19-37. N2HET Open Drain Register (HETPDR) .....	965
19-38. N2HET Pull Disable Register (HETPULDIS) .....	966
19-39. N2HET Pull Select Register (HETPSL) .....	967
19-40. Parity Control Register (HETPCR).....	968
19-41. Parity Address Register (HETPAR) .....	969
19-42. Parity Pin Register (HETPPR) .....	970
19-43. Known State on Parity Error.....	970
19-44. Suppression Filter Preload Register (HETSFPRLD) .....	971
19-45. Suppression Filter Enable Register (HETSFENA).....	972
19-46. Loop Back Pair Select Register (HETLBPSEL).....	973
19-47. Loop Back Pair Direction Register (HETLBPDIR) .....	974
19-48. NHET Pin Disable Register (HETPINDIS).....	975
19-49. Instruction Summary.....	976
19-50. FLAGS Generated by Instruction .....	976
19-51. Interrupt Capable Instructions .....	977



19-52. Arithmetic / Bitwise Logic Sub-Opcodes .....	989
19-53. Source Operand Choices .....	989
19-54. Destination Operand Choices .....	989
19-55. Execution Time for ADC, ADD, AND, OR, SBB, SUB, XOR Instructions .....	990
19-56. Move Types for ADM32 .....	996
19-57. Edge Select Encoding for APCNT .....	999
19-58. Branch Condition Encoding for BR .....	1002
19-59. DADM64 Control Field Description .....	1007
19-60. Event Encoding Format for ECNT .....	1014
19-61. Magnitude Compare Order for MCMP .....	1016
19-62. Move Type Encoding Selection .....	1019
19-63. MOV64 Control Field Descriptions .....	1023
19-64. Comparison Type Encoding Format .....	1023
19-65. Counter Type Encoding Format .....	1026
19-66. Comparison Type Encoding Format .....	1032
19-67. RADM64 Control Field Descriptions .....	1032
19-68. Step Width Encoding for SCNT .....	1039
19-69. SHIFT MODE Encoding Format .....	1041
19-70. SHIFT Condition Encoding .....	1041
19-71. Event Encoding Format for WCAP .....	1044
19-72. Event Encoding Format for WCAPE .....	1046
20-1. CPENA / TMBx Priority Rules .....	1055
20-2. Triggered Control Packets.....	1058
20-3. DCP RAM .....	1060
20-4. DCP Parity RAM.....	1061
20-5. Field Addresses of the WCAP, ECNT, PCNT Example .....	1062
20-6. 32-Bit-Transfer of Data Fields .....	1063
20-7. Destination Buffer Values .....	1063
20-8. 64-Bit-Transfer of Control Field and Data Fields .....	1063
20-9. Destination Buffer Values .....	1064
20-10. Control Register Mapping .....	1065
20-11. Global Control Register (HTU GC) Field Descriptions .....	1066
20-12. Control Packet Enable Register (HTU CPENA) Field Descriptions.....	1067
20-13. CPENA Write Results.....	1067
20-14. CPENA Read Results.....	1067
20-15. Control Packet (CP) Busy Register 0 (HTU BUSY0) Field Descriptions.....	1068
20-16. Control Packet (CP) Busy Register 1 (HTU BUSY1) Field Descriptions.....	1069
20-17. Control Packet (CP) Busy Register 2 (HTU BUSY2) Field Descriptions.....	1070
20-18. Control Packet (CP) Busy Register 3 (HTU BUSY3) Field Descriptions.....	1071
20-19. Active Control Packet and Error Register (HTU ACPE) Field Descriptions .....	1072
20-20. Request Lost and Bus Error Control Register (HTU RLBECTRL) Field Descriptions.....	1074
20-21. Buffer Full Interrupt Enable Set Register (HTU BFINTS) Field Descriptions .....	1075
20-22. Buffer Full Interrupt Enable Clear Register (HTU BFINTC) Field Descriptions .....	1076
20-23. Interrupt Mapping Register (HTU INTMAP) Field Descriptions .....	1077
20-24. Interrupt Offset Register 0 (HTU INTOFF0) Field Descriptions .....	1078
20-25. Interrupt Offset Register 1 (HTU INTOFF1) Field Descriptions .....	1080
20-26. Buffer Initialization Mode Register (HTU BIM) Field Descriptions .....	1081
20-27. Buffer Initialization .....	1081
20-28. Request Lost Flag Register (HTU RLOSTFL) Field Descriptions .....	1083



20-29. Buffer Full Interrupt Flag Register (HTU BFINTFL) Field Descriptions .....	1084
20-30. BER Interrupt Flag Register (HTU BERINTFL) Field Descriptions .....	1085
20-31. Memory Protection 1 Start Address Register (HTU MP1S) Field Descriptions .....	1086
20-32. Memory Protection 1 End Address Register (HTU MP1E) Field Descriptions .....	1087
20-33. Debug Control Register (HTU DCTRL) Field Descriptions .....	1088
20-34. Watch Point Register (HTU WPR) Field Descriptions .....	1089
20-35. Watch Mask Register (HTU WMR) Field Descriptions .....	1090
20-36. Module Identification Register (HTU ID) Field Descriptions .....	1091
20-37. Parity Control Register (HTU PCR) Field Descriptions .....	1092
20-38. Parity Address Register (HTU PAR) Field Descriptions .....	1093
20-39. Memory Protection Control and Status Register (HTU MPCS) Field Descriptions .....	1094
20-40. Memory Protection 0 Start address Register (HTU MP0S) Field Descriptions .....	1096
20-41. Memory Protection End Address Register (HTU MP0E) Field Descriptions .....	1097
20-42. Double Control Packet Memory Map .....	1098
20-43. Initial Full Address A Register (HTU IFADDRA) Field Descriptions .....	1099
20-44. Initial Full Address B Register (HTU IFADDRB) Field Descriptions .....	1100
20-45. Initial NHET Address and Control Register (HTU IHADDRCT) Field Descriptions .....	1101
20-46. Initial Transfer Count Register (HTU ITCOUNT) Field Descriptions .....	1103
20-47. Current Full Address A Register (HTU CFADDRA) Field Descriptions .....	1104
20-48. Current Full Address B Register (HTU CFADDRB) Field Descriptions .....	1105
20-49. Initial Transfer Count Register (HTU CFCOUNT) Field Descriptions .....	1106
20-50. Application Examples for Setting the Transfer Modes of CP A and B of a DCP .....	1107
21-1. Input Buffer, Output Buffer, and Pull Control Behavior as GPIO Pins .....	1116
21-2. Determining Interrupt Priority .....	1118
21-3. GIO Offset A Values and Corresponding Interrupt .....	1119
21-4. Reading the Offset Register to Determine Serviced Interrupt .....	1120
21-5. GIO Control Registers Summary .....	1122
21-6. GIO Global Control Register (GIOGCR0) Field Descriptions .....	1123
21-7. GIO Interrupt Detect Register (GIOINTDET) Field Descriptions .....	1124
21-8. GIO Interrupt Polarity Register (GIOPOL) Field Descriptions .....	1125
21-9. GIO Interrupt Enable Set Register (GIOENASET) Field Descriptions .....	1126
21-10. GIO Interrupt Enable Clear Register (GIOENACLR) Field Descriptions .....	1127
21-11. GIO Interrupt Priority Register (GIOLVLSET) Field Descriptions .....	1128
21-12. GIO Interrupt Priority Register (GIOLVLCLR) Field Descriptions .....	1129
21-13. GIO Interrupt Flag Register (GIOFLG) Field Descriptions .....	1130
21-14. GIO Offset A Register (GIOOFFA) Field Descriptions .....	1131
21-15. GIO Offset B Register (GIOOFFB) Field Descriptions .....	1132
21-16. GIO Emulation A Register (GIOEMUA) Field Descriptions .....	1133
21-17. GIO Emulation B Register (GIOEMUB) Field Descriptions .....	1134
21-18. GIO Data Direction Registers [A-B][7:0] (GIODIR[A-B][7:0]) Field Descriptions .....	1135
21-19. GIO Data Input Registers [A-B][7:0] (GIODIN[A-B][7:0]) Field Descriptions .....	1136
21-20. GIO Data Output Registers [A-B][7:0] (GIODOUT[A-B][7:0]) Field Descriptions .....	1137
21-21. GIO Data Set Registers [A-B][7:0] (GIODSET[A-B][7:0]) Field Descriptions .....	1138
21-22. GIO Data Clear Registers [A-B][7:0] (GIODCLR[A-B][7:0]) Field Descriptions .....	1139
21-23. GIO Open Drain Registers [A-B][7:0] (GIOPDR[A-B][7:0]) Field Descriptions .....	1140
21-24. GIO Pull Disable Registers [A-B][7:0] (GIOPULDIS[A-B][7:0]) Field Descriptions .....	1141
21-25. GIO Pull Select Registers [A-B][7:0] (GIOPSL[A-B][7:0]) Field Descriptions .....	1142
22-1. Parameters of the CAN Bit Time .....	1153
22-2. Message Object Field Descriptions .....	1161

22-3.	Message RAM Addressing in Debug/Suspend and RDA mode .....	1163
22-4.	Message Interface Register Sets 1 and 2 .....	1166
22-5.	Message Interface Register 3 .....	1167
22-6.	DCAN Control Registers.....	1190
22-7.	CAN Control Register Field Descriptions.....	1192
22-8.	Error and Status Register Field Descriptions .....	1195
22-9.	Error Counter Register Field Descriptions .....	1197
22-10.	Bit Timing Register Field Descriptions.....	1198
22-11.	Interrupt Register Field Descriptions.....	1199
22-12.	Test Register Field Descriptions.....	1200
22-13.	Parity Error Code Register Field Descriptions .....	1201
22-14.	Auto-Bus-On Time Register Field Descriptions .....	1202
22-15.	Transmission Request Registers Field Descriptions.....	1204
22-16.	New Data Registers Field Descriptions.....	1206
22-17.	Interrupt Pending Registers Field Descriptions.....	1208
22-18.	Message Valid Registers Field Descriptions.....	1210
22-19.	Interrupt Multiplexer Registers Field Descriptions.....	1211
22-20.	IF1/IF2 Command Register Field Descriptions .....	1213
22-21.	IF1/IF2 Mask Register Field Descriptions .....	1215
22-22.	IF1/IF2 Arbitration Register Field Descriptions .....	1216
22-23.	IF1/IF2 Message Control Register Field Descriptions .....	1218
22-24.	IF3 Observation Register Field Descriptions.....	1221
22-25.	IF3 Mask Register Field Descriptions.....	1223
22-26.	IF3 Arbitration Register Field Descriptions.....	1224
22-27.	IF3 Message Control Register Field Descriptions.....	1225
22-28.	IF3 Update Control Register Field Descriptions .....	1228
22-29.	CAN TX IO Control Register Field Descriptions.....	1229
22-30.	CAN RX IO Control Register Field Descriptions .....	1231
23-1.	Pin Configurations .....	1236
23-2.	Clocking Modes .....	1242
23-3.	Pin Mapping for SIMO Pin with MSB First .....	1248
23-4.	Pin Mapping for SOMI Pin with MSB First .....	1249
23-5.	Pin Mapping for SIMO Pin with LSB First .....	1249
23-6.	Pin Mapping for SOMI Pin with LSB First .....	1250
23-7.	SPI Registers .....	1264
23-8.	SPI Global Control Register 0 (SPIGCR0) Field Descriptions.....	1266
23-9.	SPI Global Control Register 1 (SPIGCR1) Field Descriptions.....	1267
23-10.	SPI Interrupt Register (SPIINT0) Field Descriptions.....	1269
23-11.	SPI Interrupt Level Register (SPILVL) Field Descriptions.....	1271
23-12.	SPI Flag Register (SPIFLG) Field Descriptions .....	1272
23-13.	SPI Pin Control (SPIPC0) Field Descriptions .....	1275
23-14.	SPI Pin Control Register (SPIPC1) Field Descriptions .....	1277
23-15.	SPI Pin Control Register 2 (SPIPC2) Field Descriptions.....	1279
23-16.	SPI Pin Control Register 3 (SPIPC3) Field Descriptions.....	1280
23-17.	SPI Pin Control Register 4 (SPIPC4) Field Descriptions.....	1282
23-18.	SPI Pin Control Register 5 (SPIPC5) Field Descriptions.....	1284
23-19.	SPI Pin Control Register 6 (SPIPC6) Field Descriptions.....	1286
23-20.	SPI Pin Control Register 7 (SPIPC7) Field Descriptions.....	1288
23-21.	SPI Pin Control Register 8 (SPIPC8) Field Descriptions.....	1290

23-22. SPI Transmit Data Register 0 (SPIDAT0) Field Descriptions .....	1292
23-23. SPI Transmit Data Register 1 (SPIDAT1) Field Descriptions .....	1293
23-24. SPI Receive Buffer Register (SPIBUF) Field Descriptions.....	1294
23-25. SPI Emulation Register (SPIEMU) Field Descriptions.....	1296
23-26. SPI Delay Register (SPIDELAY) Field Descriptions .....	1297
23-27. SPI Default Chip Select Register (SPIDEF) Field Descriptions .....	1300
23-28. SPI Data Format Registers (SPIFMT[3:0]) Field Descriptions .....	1301
23-29. Transfer Group Interrupt Vector 0 (INTVECT0) .....	1303
23-30. Transfer Group Interrupt Vector 1 (INTVECT1) .....	1305
23-31. SPI Parallel/Modulo Mode Control Register (SPIPMCTRL) Field Descriptions .....	1307
23-32. Multi-buffer Mode Enable Register (MIBSPIE) Field Descriptions .....	1310
23-33. TG Interrupt Enable Set Register (TGITENST) Field Descriptions .....	1311
23-34. TG Interrupt Enable Clear Register (TGITENCR) Field Descriptions .....	1312
23-35. Transfer Group Interrupt Level Set Register (TGITLVST) Field Descriptions .....	1313
23-36. Transfer Group Interrupt Level Clear Register (TGITLVCR) Field Descriptions .....	1314
23-37. Transfer Group Interrupt Level Clear Register (TGITLVCR) Field Descriptions .....	1315
23-38. Tick Count Register (TICKCNT) Field Descriptions.....	1316
23-39. Last TG End Pointer (LTGPEND) Field Descriptions.....	1317
23-40. TG Control Registers (TGxCTRL) Field Descriptions .....	1318
23-41. DMA Channel Control Register (DMAxCTRL) Field Descriptions .....	1321
23-42. MibSPI DMAxCOUNT Register (ICOUNT) Field Descriptions .....	1323
23-43. MibSPI DMA Large Count Register (DMACNTLEN) Field Descriptions .....	1324
23-44. Multi-buffer RAM Uncorrectable Parity Error Control Register (UERRCTRL) Field Descriptions.....	1325
23-45. Multi-buffer RAM Uncorrectable Parity Error Status Register (UERRSTAT) Field Descriptions.....	1326
23-46. RXRAM Uncorrectable Parity Error Address Register (UERRADDR1) Field Descriptions.....	1327
23-47. TXRAM Uncorrectable Parity Error Address Register (UERRADDR0) Field Descriptions .....	1328
23-48. RXRAM Overrun Buffer Address Register (RXOVRN_BUF_ADDR) Field Descriptions.....	1329
23-49. I/O-Loopback Test Control Register (IOLPBKTSTCR) Field Descriptions.....	1330
23-50. Multi-buffer RAM Register Summary .....	1334
23-51. Multi-buffer RAM Transmit Data Register Field Descriptions.....	1335
23-52. Multi-buffer Receive Buffer Register Field Descriptions.....	1337
23-53. Timing parameters of SPI/MibSPI pins in Master Mode .....	1344
23-54. Timing parameters of SPI/MibSPI pins in Slave Mode. ....	1346
24-1. Superfractional Bit Modulation for SCI Mode (Normal Configuration) .....	1355
24-2. Superfractional Bit Modulation for SCI Mode (Maximum Configuration) .....	1356
24-3. SCI Mode (Minimum Configuration) .....	1356
24-4. SCI/LIN Interrupts .....	1363
24-5. Response Length Info Using IDBYTE Field Bits [5:4] for LIN Standards Earlier than 1.3 .....	1370
24-6. Response Length with SCIFORMAT[18:16] Programming .....	1370
24-7. Superfractional Bit Modulation for LIN Master Mode and Slave Mode .....	1372
24-8. Timeout Values in T <sub>bit</sub> Units .....	1379
24-9. SCI/LIN Control Registers Summary .....	1392
24-10. SCI Global Control Register 0 (SCIGCR0) Field Descriptions .....	1393
24-11. SCI Global Control Register 1 (SCIGCR1) Field Descriptions .....	1394
24-12. SCI Receiver Status Flags .....	1397
24-13. SCI Transmitter Status Flags .....	1397
24-14. SCI Global Control Register 2 (SCIGCR2) Field Descriptions .....	1398
24-15. SCI Set Interrupt Register (SCISSETINT) Field Descriptions.....	1400
24-16. SCI Clear Interrupt Register (SCICLEARINT) Field Descriptions.....	1403

24-17. SCI Set Interrupt Level Register (SCISSETINTLVL) Field Descriptions .....	1407
24-18. SCI Clear Interrupt Level Register (SCICLEARINTLVL) Field Descriptions.....	1410
24-19. SCI Flags Register (SCIFLR) Field Descriptions .....	1413
24-20. SCI Interrupt Vector Offset 0 (SCIINTVECT0) Field Descriptions .....	1420
24-21. SCI Interrupt Vector Offset 1 (SCIINTVECT1) Field Descriptions .....	1421
24-22. SCI Format Control Register (SCIFORMAT) Field Descriptions .....	1422
24-23. Baud Rate Selection Register (BRS) Field Descriptions .....	1423
24-24. Comparative Baud Values for Different P Values, Asynchronous Mode.....	1423
24-25. Receiver Emulation Data Buffer (SCIED) Field Descriptions .....	1425
24-26. Receiver Data Buffer (SCIRD) Field Descriptions .....	1426
24-27. Transmit Data Buffer Register (SCITD) Field Descriptions .....	1427
24-28. SCI Pin I/O Control Register 0 (SCIPIO0) Field Descriptions .....	1428
24-29. SCI Pin I/O Control Register 1 (SCIPIO1) Field Descriptions .....	1429
24-30. LINTX Pin Control .....	1429
24-31. LINRX Pin Control .....	1429
24-32. SCI Pin I/O Control Register 2 (SCIPIO2) Field Descriptions .....	1430
24-33. SCI Pin I/O Control Register 3 (SCIPIO3) Field Descriptions .....	1431
24-34. SCI Pin I/O Control Register 4 (SCIPIO4) Field Descriptions .....	1432
24-35. SCI Pin I/O Control Register 5 (SCIPIO5) Field Descriptions .....	1433
24-36. SCI Pin I/O Control Register 6 (SCIPIO6) Field Descriptions .....	1434
24-37. SCI Pin I/O Control Register 7 (SCIPIO7) Field Descriptions .....	1435
24-38. SCI Pin I/O Control Register 8 (SCIPIO8) Field Descriptions .....	1436
24-39. LIN Compare Register (LINCOMPARE) Field Descriptions .....	1437
24-40. LIN Receive Buffer 0 Register (LINRD0) Field Descriptions.....	1438
24-41. LIN Receive Buffer 1 Register (RD1) Field Descriptions.....	1439
24-42. LIN Mask Register (LINMASK) Field Descriptions .....	1440
24-43. LIN Identification Register (LINID) Field Descriptions .....	1441
24-44. LIN Transmit Buffer 0 Register (LINTD0) Field Descriptions .....	1442
24-45. LIN Transmit Buffer 1 Register (LINTD1) Field Descriptions .....	1443
24-46. Maximum Baud Rate Selection Register (MBRS) Field Descriptions .....	1444
24-47. Input/Output Error Enable Register (IODFTCTRL) Field Descriptions .....	1445
24-48. Input Buffer, Output Buffer, and Pull Control Behavior as GPIO Pins .....	1448
25-1. SCI Interrupts .....	1459
25-2. DMA and Interrupt Requests in Multiprocessor Modes .....	1460
25-3. SCI Control Registers Summary .....	1465
25-4. SCI Global Control Register 0 (SCIGCR0) Field Descriptions .....	1466
25-5. SCI Global Control Register 1 (SCIGCR1) Field Descriptions .....	1467
25-6. SCI Receiver Status Flags .....	1469
25-7. SCI Transmitter Status Flags .....	1469
25-8. SCI Set Interrupt Register (SCISSETINT) Field Descriptions .....	1470
25-9. SCI Clear Interrupt Register (SCICLEARINT) Field Descriptions.....	1472
25-10. SCI Set Interrupt Level Register (SCISSETINTLVL) Field Descriptions .....	1474
25-11. SCI Clear Interrupt Level Register (SCICLEARINTLVL) Field Descriptions.....	1476
25-12. SCI Flags Register (SCIFLR) Field Descriptions.....	1478
25-13. SCI Interrupt Vector Offset 0 (SCIINTVECT0) Field Descriptions .....	1482
25-14. SCI Interrupt Vector Offset 1 (SCIINTVECT1) Field Descriptions .....	1483
25-15. SCI Format Control Register (SCIFORMAT) Field Descriptions .....	1484
25-16. Baud Rate Selection Register (BRS) Field Descriptions .....	1485
25-17. Comparative Baud Values for Different P Values, Asynchronous Mode .....	1485

25-18. Receiver Emulation Data Buffer (SCIED) Field Descriptions .....	1487
25-19. Receiver Data Buffer (SCIRD) Field Descriptions .....	1488
25-20. Transmit Data Buffer Register (SCITD) Field Descriptions .....	1489
25-21. SCI Pin I/O Control Register 0 (SCIPIO0) Field Descriptions .....	1490
25-22. SCI Pin I/O Control Register 1 (SCIPIO1) Field Descriptions .....	1491
25-23. SCITX Pin Control .....	1491
25-24. SCIRX Pin Control .....	1491
25-25. SCI Pin I/O Control Register 2 (SCIPIO2) Field Descriptions .....	1492
25-26. SCI Pin I/O Control Register 3 (SCIPIO3) Field Descriptions .....	1493
25-27. SCI Pin I/O Control Register 4 (SCIPIO4) Field Descriptions .....	1494
25-28. SCI Pin I/O Control Register 5 (SCIPIO5) Field Descriptions .....	1495
25-29. SCI Pin I/O Control Register 6 (SCIPIO6) Field Descriptions .....	1496
25-30. SCI Pin I/O Control Register 7 (SCIPIO7) Field Descriptions .....	1497
25-31. SCI Pin I/O Control Register 8 (SCIPIO8) Field Descriptions .....	1498
25-32. Input/Output Error Enable Register (IODFTCTRL) Field Descriptions .....	1499
25-33. Input Buffer, Output Buffer, and Pull Control Behavior as GPIO Pins .....	1502
26-1. Ways to Generate a NACK Bit .....	1511
26-2. Interrupt Requests Generated by I2C Module .....	1517
26-3. I2C Control Register Summary .....	1520
26-4. I2C Own Address Manager Register (I2COAR) Field Descriptions .....	1521
26-5. Correct Mode for Bits OA.9:0 .....	1521
26-6. I2C Interrupt Mask Register (I2CIMR) Field Descriptions .....	1522
26-7. I2C Status Register (I2CSTR) Field Descriptions .....	1523
26-8. I2C Clock Divider Low Register (I2CCKL) Field Descriptions .....	1526
26-9. I2C Clock Control High Register (I2CCKH) Field Descriptions .....	1527
26-10. I2C Data Count Register (I2CCNT) Field Descriptions .....	1528
26-11. I2C Data Receive Register (I2CDRR) Field Descriptions .....	1529
26-12. I2C Slave Address Register (I2CSAR) Field Descriptions .....	1530
26-13. Correct Modes for Bits SA(9–0) .....	1530
26-14. I2C Data Transmit Register (I2CDXR) Field Descriptions .....	1531
26-15. I2C Mode Register (I2CMDR) Field Descriptions .....	1532
26-16. I2C Module Condition, Bus Activity, and Mode .....	1534
26-17. I2C Module Operating Modes .....	1534
26-18. Number of Bits Sent on Bus .....	1534
26-19. I2C Interrupt Vector Register (I2CIVR) Field Descriptions .....	1535
26-20. Interrupt Codes for INTCODE Bit .....	1535
26-21. I2C Extended Mode Register (I2CEMDR) Field Descriptions .....	1536
26-22. I2C Prescale Register (I2CPSC) Field Descriptions .....	1537
26-23. I2C Peripheral ID Register 1 (I2CPID1) Field Descriptions .....	1538
26-24. I2C Peripheral ID Register 2 (I2CPID2) Field Descriptions .....	1539
26-25. I2C DMA Control Register (I2CDMACR) Field Descriptions .....	1540
26-26. I2C Pin Function Register (I2CPFNC) Field Descriptions .....	1541
26-27. I2C Pin Direction Register (I2CPDIR) Field Descriptions .....	1542
26-28. I2C Data Input Register (I2CDIN) Field Descriptions .....	1543
26-29. I2C Data Output Register (I2CDOUT) Field Descriptions .....	1544
26-30. I2C Data Set Register (I2CDSET) Field Descriptions .....	1545
26-31. I2C Data Clear Register (I2CDSET) Field Descriptions .....	1546
26-32. I2C Pin Open Drain Register (I2CPDR) Field Descriptions .....	1547
26-33. I2C Pull Disable Register (I2CDCLR) Field Descriptions .....	1548



26-34. I2C Pull Disable Register (I2CDCLR) Field Descriptions .....	1549
26-35. Input Buffer, Output Buffer, and Pull Control Behavior as GPIO Pins .....	1549
26-36. I2C Pins Slew Rate Select Register (I2CSRS) Field Descriptions.....	1550
27-1. EMAC and MDIO Signals for MII Interface.....	1558
27-2. EMAC and MDIO Signals for RMII Interface.....	1559
27-3. MDIO Multiplexing Control .....	1560
27-4. MII / RMII Multiplexing Control.....	1560
27-5. Ethernet Frame Description .....	1561
27-6. Basic Descriptor Description .....	1563
27-7. Receive Frame Treatment Summary .....	1588
27-8. Middle of Frame Overrun Treatment.....	1589
27-9. Emulation Control .....	1599
27-10. EMAC Control Module Registers.....	1600
27-11. EMAC Control Module Revision ID Register (REVID) Field Descriptions .....	1601
27-12. EMAC Control Module Software Reset Register (SOFTRESET) .....	1602
27-13. EMAC Control Module Interrupt Control Register (INTCONTROL).....	1603
27-14. EMAC Control Module Receive Threshold Interrupt Enable Register (C0RXTHRESHEN) .....	1604
27-15. EMAC Control Module Receive Interrupt Enable Register (C0RXEN) .....	1605
27-16. EMAC Control Module Transmit Interrupt Enable Register (C0TXEN).....	1606
27-17. EMAC Control Module Miscellaneous Interrupt Enable Register (C0MISCEN).....	1607
27-18. EMAC Control Module Receive Threshold Interrupt Status Register (C0RXTHRESHSTAT).....	1608
27-19. EMAC Control Module Receive Interrupt Status Register (C0RXSTAT).....	1609
27-20. EMAC Control Module Transmit Interrupt Status Register (C0TXSTAT) .....	1610
27-21. EMAC Control Module Miscellaneous Interrupt Status Register (C0MISCSTAT) .....	1611
27-22. EMAC Control Module Receive Interrupts Per Millisecond Register (C0RXIMAX) .....	1612
27-23. EMAC Control Module Transmit Interrupts Per Millisecond Register (C0TXIMAX) .....	1613
27-24. Management Data Input/Output (MDIO) Registers .....	1614
27-25. MDIO Revision ID Register (REVID) Field Descriptions .....	1615
27-26. MDIO Control Register (CONTROL) Field Descriptions .....	1616
27-27. PHY Acknowledge Status Register (ALIVE) Field Descriptions.....	1617
27-28. PHY Link Status Register (LINK) Field Descriptions .....	1618
27-29. MDIO Link Status Change Interrupt (Unmasked) Register (LINKINTRAW) Field Descriptions.....	1619
27-30. MDIO Link Status Change Interrupt (Masked) Register (LINKINTMASKED) Field Descriptions .....	1620
27-31. MDIO User Command Complete Interrupt (Unmasked) Register (USERINTRAW) Field Descriptions.....	1621
27-32. MDIO User Command Complete Interrupt (Masked) Register (USERINTMASKED) Field Descriptions ...	1622
27-33. MDIO User Command Complete Interrupt Mask Set Register (USERINTMASKSET) Field Descriptions .	1623
27-34. MDIO User Command Complete Interrupt Mask Clear Register (USERINTMASKCLEAR) Field Descriptions .....	1624
27-35. MDIO User Access Register 0 (USERACCESS0) Field Descriptions.....	1625
27-36. MDIO User PHY Select Register 0 (USERPHYSEL0) Field Descriptions .....	1626
27-37. MDIO User Access Register 1 (USERACCESS1) Field Descriptions.....	1627
27-38. MDIO User PHY Select Register 1 (USERPHYSEL1) Field Descriptions .....	1628
27-39. Ethernet Media Access Controller (EMAC) Registers .....	1629
27-40. Transmit Revision ID Register (TXREVID) Field Descriptions .....	1633
27-41. Transmit Control Register (TXCONTROL) Field Descriptions.....	1634
27-42. Transmit Teardown Register (TXTEARDOWN) Field Descriptions.....	1635
27-43. Receive Revision ID Register (RXREVID) Field Descriptions.....	1636
27-44. Receive Control Register (RXCONTROL) Field Descriptions .....	1637
27-45. Receive Teardown Register (RXTEARDOWN) Field Descriptions .....	1638



27-46. Transmit Interrupt Status (Unmasked) Register (TXINTSTATRAW) Field Descriptions .....	1639
27-47. Transmit Interrupt Status (Masked) Register (TXINTSTATMASKED) Field Descriptions .....	1640
27-48. Transmit Interrupt Mask Set Register (TXINTMASKSET) Field Descriptions .....	1641
27-49. Transmit Interrupt Mask Clear Register (TXINTMASKCLEAR) Field Descriptions .....	1642
27-50. MAC Input Vector Register (MACINVECTOR) Field Descriptions .....	1643
27-51. MAC End Of Interrupt Vector Register (MACEOIVECTOR) Field Descriptions.....	1644
27-52. Receive Interrupt Status (Unmasked) Register (RXINTSTATRAW) Field Descriptions .....	1645
27-53. Receive Interrupt Status (Masked) Register (RXINTSTATMASKED) Field Descriptions.....	1646
27-54. Receive Interrupt Mask Set Register (RXINTMASKSET) Field Descriptions .....	1647
27-55. Receive Interrupt Mask Clear Register (RXINTMASKCLEAR) Field Descriptions .....	1648
27-56. MAC Interrupt Status (Unmasked) Register (MACINTSTATRAW) Field Descriptions .....	1649
27-57. MAC Interrupt Status (Masked) Register (MACINTSTATMASKED) Field Descriptions .....	1650
27-58. MAC Interrupt Mask Set Register (MACINTMASKSET) Field Descriptions .....	1651
27-59. MAC Interrupt Mask Clear Register (MACINTMASKCLEAR) Field Descriptions.....	1652
27-60. Receive Multicast/Broadcast/Promiscuous Channel Enable Register (RXMBPENABLE) Field Descriptions .....	1653
27-61. Receive Unicast Enable Set Register (RXUNICASTSET) Field Descriptions.....	1656
27-62. Receive Unicast Clear Register (RXUNICASTCLEAR) Field Descriptions.....	1657
27-63. Receive Maximum Length Register (RXMAXLEN) Field Descriptions .....	1658
27-64. Receive Buffer Offset Register (RXBUFFEROFFSET) Field Descriptions .....	1659
27-65. Receive Filter Low Priority Frame Threshold Register (RXFILTERLOWTHRESH) Field Descriptions .....	1660
27-66. Receive Channel <i>n</i> Flow Control Threshold Register (RX <i>n</i> FLOWTHRESH) Field Descriptions .....	1661
27-67. Receive Channel <i>n</i> Free Buffer Count Register (RX <i>n</i> FREEBUFFER) Field Descriptions.....	1662
27-68. MAC Control Register (MACCONTROL) Field Descriptions.....	1663
27-69. MAC Status Register (MACSTATUS) Field Descriptions .....	1665
27-70. Emulation Control Register (EMCONTROL) Field Descriptions .....	1667
27-71. FIFO Control Register (FIFOCONTROL) Field Descriptions .....	1668
27-72. MAC Configuration Register (MACCONFIG) Field Descriptions .....	1669
27-73. Soft Reset Register (SOFTRESET) Field Descriptions .....	1670
27-74. MAC Source Address Low Bytes Register (MACSRCADDRLO) Field Descriptions.....	1671
27-75. MAC Source Address High Bytes Register (MACSRCADDRHI) Field Descriptions .....	1672
27-76. MAC Hash Address Register 1 (MACHASH1) Field Descriptions .....	1673
27-77. MAC Hash Address Register 2 (MACHASH2) Field Descriptions .....	1674
27-78. Back Off Test Register (BOFFTEST) Field Descriptions.....	1675
27-79. Transmit Pacing Algorithm Test Register (TPACETEST) Field Descriptions .....	1676
27-80. Receive Pause Timer Register (RXPAUSE) Field Descriptions .....	1677
27-81. Transmit Pause Timer Register (TXPAUSE) Field Descriptions .....	1678
27-82. MAC Address Low Bytes Register (MACADDRLO) Field Descriptions .....	1679
27-83. MAC Address High Bytes Register (MACADDRHI) Field Descriptions .....	1680
27-84. MAC Index Register (MACINDEX) Field Descriptions.....	1681
27-85. Transmit Channel <i>n</i> DMA Head Descriptor Pointer Register (TX <i>n</i> HDP) Field Descriptions .....	1682
27-86. Receive Channel <i>n</i> DMA Head Descriptor Pointer Register (RX <i>n</i> HDP) Field Descriptions.....	1683
27-87. Transmit Channel <i>n</i> Completion Pointer Register (TX <i>n</i> CP) Field Descriptions .....	1684
27-88. Receive Channel <i>n</i> Completion Pointer Register (RX <i>n</i> CP) Field Descriptions .....	1685
28-1. USB Host / Device Interface Signal Multiplexing and Control .....	1698
28-2. USB Host Controller Registers .....	1699
28-3. OHCI Revision Number Register (HCREVISION) Bit Field Descriptions .....	1700
28-4. HC Operating Mode Register (HCCONTROL) Bit Field Descriptions .....	1701
28-5. HC Command and Status Register (HCCOMMANDSTATUS) Bit Field Descriptions.....	1702

28-6.	HC Interrupt Status Register (HCINTERRUPTSTATUS) Bit Field Descriptions .....	1703
28-7.	HC Interrupt Enable Register (HCINTERRUPTENABLE) Bit Field Descriptions .....	1704
28-8.	HC Interrupt Disable Register (HCINTERRUPTDISABLE) Bit Field Descriptions .....	1705
28-9.	HC HCCA Address Register (HCHCCA) Bit Field Descriptions.....	1706
28-10.	HC Current Periodic Register (HCPERIODCURRENTED) Bit Field Descriptions .....	1707
28-11.	HC Head Control Register (HCCONTROLHEADED) Bit Field Descriptions .....	1707
28-12.	HC Current Control Register (HCCONTROLCURRENTED) Bit Field Descriptions .....	1707
28-13.	HC Head Bulk Register (HCBULKHEADED) Bit Field Descriptions .....	1708
28-14.	HC Current Bulk Register (HCBULKCURRENTED) Bit Field Descriptions.....	1708
28-15.	HC Head Done Register (HCDONEHEAD) Bit Field Descriptions.....	1709
28-16.	HC Frame Interval Register (HCFMINTERVAL) Bit Field Descriptions .....	1709
28-17.	HC Frame Remaining Register (HCFMREMAINING) Bit Field Descriptions .....	1710
28-18.	HC Frame Number Register (HCFMNUMBER) Bit Field Descriptions .....	1710
28-19.	HC Periodic Start Register (HCPERIODICSTART) Bit Field Descriptions .....	1711
28-20.	HC Low-Speed Threshold Register (HCLSTHRESHOLD) Bit Field Descriptions .....	1711
28-21.	HC Root Hub A Register (HCRHDESCRIPTORA) Field Descriptions .....	1712
28-22.	HC Root Hub B Register (HCRHDESCRIPTORB) Bit Field Descriptions .....	1713
28-23.	HC Root Hub Status Register (HCRHSTATUS) Bit Field Descriptions .....	1714
28-24.	HC Port 0 Status and Control Register (HCRHPORTSTATUS0) Field Descriptions.....	1715
28-25.	HC Port 1 Status and Control Register (HCRHPORTSTATUS1) Field Descriptions.....	1717
28-26.	Host UE Address Register (HOSTUEADDR) Field Descriptions .....	1719
28-27.	Host UE Status Register (HOSTUESTATUS) Bit Field Descriptions .....	1719
28-28.	Host Time-out Control Register (HOSTTIMEOUTCTRL) Bit Field Descriptions .....	1720
28-29.	Host Revision Register (HOSTREVISION) Field Descriptions .....	1720
28-30.	USB Device Controller Registers.....	1723
28-31.	Revision Register (REV) Bit Field Descriptions .....	1724
28-32.	Endpoint Selection Register (EP_NUM) Bit Field Descriptions .....	1724
28-33.	Data Register (DATA) Bit Field Descriptions .....	1725
28-34.	Control Register (CTRL) Bit Field Descriptions .....	1726
28-35.	Status Register (STAT_FLG) Bit Field Descriptions .....	1727
28-36.	Receive FIFO Status Register (RXFSTAT) Bit Field Descriptions .....	1730
28-37.	System Configuration Register 1 (SYSCON1) Bit Field Descriptions .....	1730
28-38.	Little Endian and Big Endian Formats .....	1731
28-39.	System Configuration Register 2 (SYSCON2) Bit Field Descriptions .....	1731
28-40.	Device Status Register (DEVSTAT) Bit Field Descriptions .....	1733
28-41.	Start of Frame Register (SOF) Bit Field Descriptions .....	1734
28-42.	Interrupt Enable Register (IRQ_EN) Bit Field Descriptions .....	1735
28-43.	DMA Interrupt Enable Register (DMA_IRQ_EN) Bit Field Descriptions.....	1736
28-44.	Interrupt Source Register (IRQ_SRC) Bit Field Descriptions.....	1737
28-45.	Non-ISO Endpoint Interrupt Status Register (EPN_STAT) Bit Field Descriptions .....	1739
28-46.	Non-ISO DMA Interrupt Status Register (DMAN_STAT) Bit Field Descriptions .....	1739
28-47.	DMA Receive Channels Configuration Register (RXDMA_CFG) Bit Field Descriptions.....	1740
28-48.	DMA Transmit Channels Configuration Register (TXDMA_CFG) Bit Field Descriptions .....	1742
28-49.	DMA FIFO Data Register (DATA_DMA) Bit Field Descriptions .....	1743
28-50.	Transmit DMA Control Register n (TXDMA <sub>n</sub> ) Bit Field Descriptions .....	1743
28-51.	Receive DMA Control Register n (RXDMA <sub>n</sub> ) Bit Field Descriptions.....	1744
28-52.	Endpoint 0 Configuration Register (EP0) Bit Field Descriptions .....	1745
28-53.	Receive Endpoint n Configuration Register (EP <sub>n</sub> _RX) Bit Field Descriptions.....	1746
28-54.	Transmit Endpoint n Configuration Register (EP <sub>n</sub> _TX) Bit Field Descriptions .....	1747

28-55. Autodecoded Versus Non-Autodecoded Control Requests .....	1765
28-56. USB Device Controller Interrupt Type by Endpoint Type .....	1795
28-57. Signaling Between USB Controller and Unidirectional USB Transceiver Using DAT/SE0 Signaling .....	1809
29-1. Encoding of Destination Bits in Trace Mode Packet Format .....	1816
29-2. Encoding of Status Bits in Trace Mode Packet Format .....	1816
29-3. Encoding of Write Size in Packet Format .....	1816
29-4. Number of Clock Cycles per Packet .....	1817
29-5. Pins Used for Data Communication.....	1817
29-6. Signal Description .....	1818
29-7. DMM Registers Summary .....	1820
29-8. DMM Global Control Register (DMMGLBCTRL) Field Descriptions.....	1821
29-9. DMM Interrupt Set Register (DMMINTSET) Field Descriptions .....	1823
29-10. DMM Interrupt Clear Register (DMMINTCLR) Field Descriptions .....	1827
29-11. DMM Interrupt Level Register (DMMINTLVL) Field Descriptions .....	1832
29-12. DMM Interrupt Flag Register (DMMINTFLG) Field Descriptions.....	1834
29-13. DMM Interrupt Offset 1 Register (DMMOFF1) Field Descriptions .....	1838
29-14. DMM Interrupt Offset 2 Register (DMMOFF1) Field Descriptions .....	1839
29-15. DMM Direct Data Mode Destination Register (DMMDDMDEST) Field Descriptions.....	1840
29-16. DMM Direct Data Mode Blocksize Register (DMMDDMBL) Field Descriptions.....	1841
29-17. DMM Direct Data Mode Pointer Register (DMMDDMPT) Field Descriptions .....	1842
29-18. DMM Direct Data Mode Interrupt Pointer Register (DMMINTPT) Field Descriptions .....	1843
29-19. DMM Destination x Region 1 (DMMDESTxREG1) Field Descriptions .....	1844
29-20. DMM Destination x Blocksize 1 (DMMDESTxBL1) Field Descriptions .....	1845
29-21. DMM Destination x Region 2 (DMMDESTxREG2) Field Descriptions .....	1846
29-22. DMM Destination x Blocksize 2 (DMMDESTxBL2) Field Descriptions .....	1847
29-23. DMM Pin Control 0 (DMMPC0) Field Descriptions .....	1848
29-24. DMM Pin Control 1 (DMMPC1) Field Descriptions .....	1850
29-25. DMM Pin Control 2 (DMMPC2) Field Descriptions .....	1852
29-26. DMM Pin Control 3 (DMMPC3) Field Descriptions .....	1853
29-27. DMM Pin Control 4 (DMMPC4) Field Descriptions .....	1855
29-28. DMM Pin Control 5 (DMMPC5) Field Descriptions .....	1857
29-29. DMM Pin Control 6 (DMMPC6) Field Descriptions .....	1859
29-30. DMM Pin Control 7 (DMMPC7) Field Descriptions .....	1861
29-31. DMM Pin Control 8 (DMMPC8) Field Descriptions .....	1863
30-1. Encoding of RAM Bits in Trace Mode Packet Format .....	1867
30-2. Encoding of Status Bits in Trace Mode Packet Format .....	1868
30-3. Encoding of SIZE bits in Trace Mode Packet Format .....	1868
30-4. Encoding of REG in Trace Mode Packet Format .....	1868
30-5. Number of Transfers/Packet .....	1868
30-6. Module Registers .....	1874
30-7. RTP Global Control Register (RTPGLBCTRL) Field Descriptions.....	1875
30-8. FIFO Corresponding Addresses.....	1877
30-9. Pins Used for Data Communication.....	1877
30-10. RTP Trace Enable Register (RTPTRENA) Field Descriptions .....	1878
30-11. RTP Global Status Register (RTPGSR) [offset = 08h] Field Descriptions.....	1879
30-12. RTP RAM 1 Trace Region [1:2] Register (RTPRAM1REG[1:2]) Field Descriptions.....	1881
30-13. RTP RAM 2 Trace Region [1:2] Register (RTPRAM2REG[1:2]) Field Descriptions.....	1882
30-14. RTP Peripheral Trace Region [1:2] Register (RTPPERREG[1:2]) Field Descriptions.....	1883
30-15. RTP Direct Data Mode Write Register (RTPDDMW) [offset = 2Ch] Field Descriptions.....	1884

---

30-16. RTP Pin Control 0 Register (RTPPC0) [offset = 34h] Field Descriptions .....	1885
30-17. RTP Pin Control 1 Register (RTPPC1) Field Descriptions.....	1886
30-18. RTP Pin Control 2 Register (RTPPC2) [offset = 3Ch] Field Descriptions.....	1887
30-19. RTP Pin Control 3 Register (RTPPC3) Field Descriptions.....	1888
30-20. RTP Pin Control 4 Register (RTPPC4) Field Descriptions.....	1889
30-21. RTP Pin Control 5 Register (RTPPC5) [offset = 48h] Field Descriptions .....	1890
30-22. RTP Pin Control 6 Register (RTPPC6) Field Descriptions.....	1891
30-23. RTP Pin Control 7 Register (RTPPC7) Field Descriptions.....	1893
30-24. RTP Pin Control 8 Register (RTPPC8) [offset = 54h] Field Descriptions .....	1894



## ***Read This First***

---

---

---

### **About This Manual**

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

This TRM has been designated *Preliminary* because the documentation is in the formative or design phase of development. Texas Instruments reserves the right to change this TRM without notice. Visit the Texas Instruments website at [www.ti.com](http://www.ti.com) to determine the latest version of this TRM.

Revision History will not be provided for this TRM as long as it is in the *Preliminary* phase of development.

### **Related Documentation From Texas Instruments**

For a complete listing of related documentation and development-support tools, visit the Texas Instruments website at [www.ti.com](http://www.ti.com).





## ***Introduction***

---

---

---

Topic	Page
1.1 <b>Designed for Safety Applications</b> .....	<b>86</b>
1.2 <b>Family Description</b> .....	<b>86</b>
1.3 <b>Endianism Considerations</b> .....	<b>89</b>

## 1.1 Designed for Safety Applications

The RM48x device architecture has been designed from the ground up to simplify development of functionally safe systems. The basic architectural concept is known as a safe island approach. Power, clock, reset, and basic processing function are protected to a high level of diagnostic coverage in hardware. Some of the key features of the safe island region are:

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

The RM48x device architecture also includes many features to simplify diagnostics of remaining logic such as:

- Continuous parity diagnostics on all peripheral memories
- Analog and digital loopback to test for shorts on I/O
- HW self-test and diagnostics on the ADC module to check integrity of both analog inputs and the ADC core conversion function
- A DMA driven hardware engine for the background calculation of CRC signatures during data transfers
- A centralized error reporting function including a status output pin to enable external monitoring of the TMS570 status

## 1.2 Family Description

The RM48x integrates the ARM® Cortex™-R4F Floating Point CPU which offers an efficient 1.6 DMIPS/MHz and has configurations which can run up to 220MHz providing up to 352 DMIPS. The device supports the little-endian [LE] format.

The RM48x has up to 3MB integrated Flash and up to 256KB data RAM configurations with single bit error correction and double bit error detection. The flash memory on this device is a nonvolatile, electrically erasable and programmable memory implemented with a 64-bit-wide data bus interface. The flash operates on a 3.3V supply input (same level as I/O supply) for all read, program and erase operations. When in pipeline mode, the flash operates with a system clock frequency of up to 220MHz. The SRAM supports single-cycle read/write accesses in byte, halfword, and word modes.

The RM48x device features peripherals for real-time control-based applications, including two Next Generation High End Timer (N2HET) timing coprocessors with up to 40 total IO terminals and a 12-bit A to D converter supporting up to 24 inputs.

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

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

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

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

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

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

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

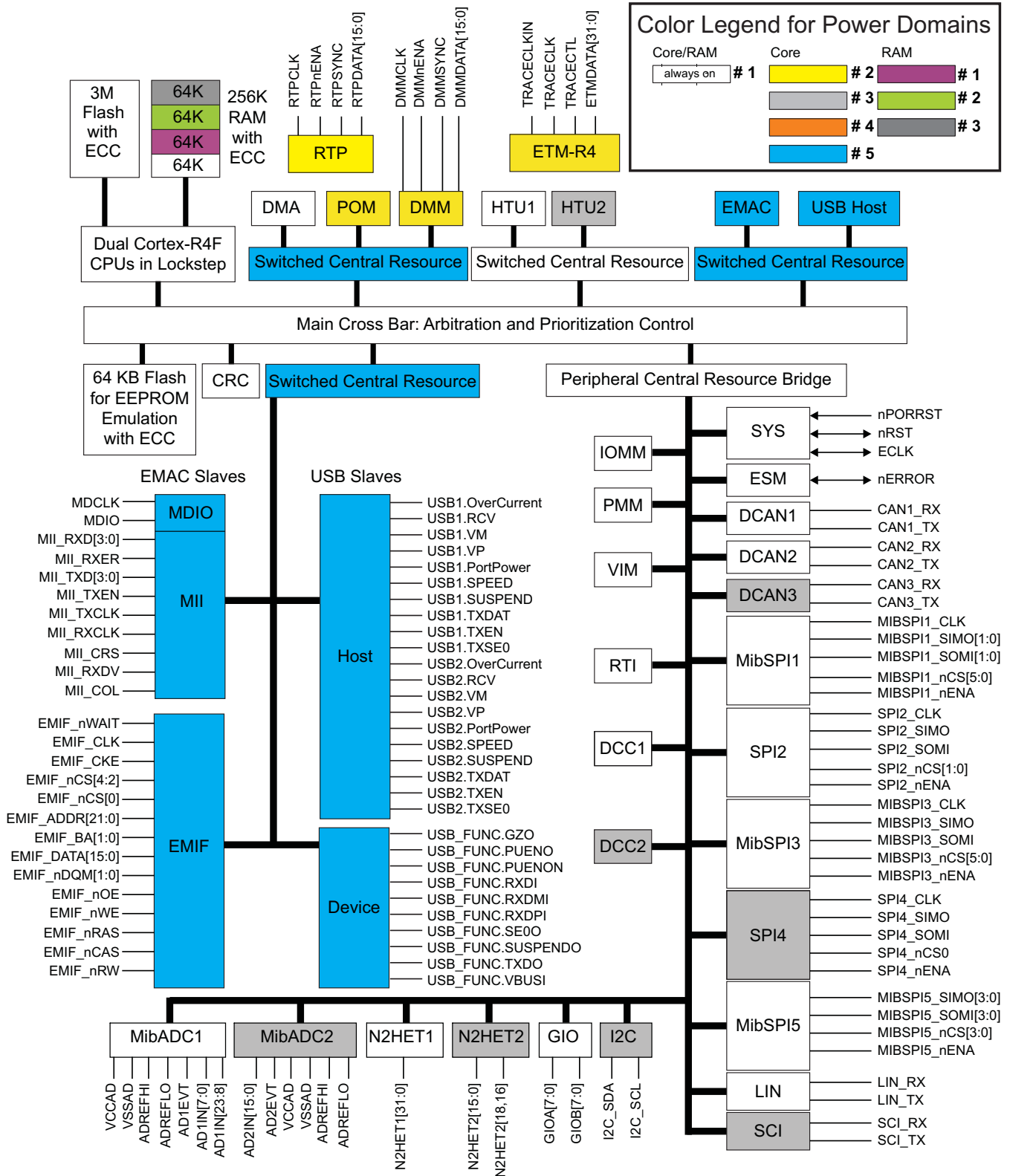
The Error Signaling Module (ESM) monitors all device errors and determines whether an interrupt or external Error pin/ball is triggered when a fault is detected. The nERROR can be monitored externally as an indicator of a fault condition in the microcontroller.

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

Several interfaces are implemented to enhance the debugging capabilities of application code. In addition to the built in ARM Cortex™-R4F CoreSight™ debug features. An External Trace Macrocell (ETM) provides instruction and data trace of program execution. For instrumentation purposes, a RAM Trace Port Module (RTP) is implemented to support high-speed tracing of RAM and peripheral accesses by the CPU or any other master. A Data Modification Module (DMM) gives the ability to write external data into the device memory. Both the RTP and DMM have no or only minimum impact on the program execution time of the application code. A Parameter Overlay Module (POM) can re-route Flash accesses to internal memory or to the EMIF, thus avoiding the re-programming steps necessary for parameter updates in Flash.

With integrated safety features and a wide choice of communication and control peripherals, the RM48x is an ideal solution for high performance real time control applications with safety critical requirements.

Figure 1-1. Block Diagram

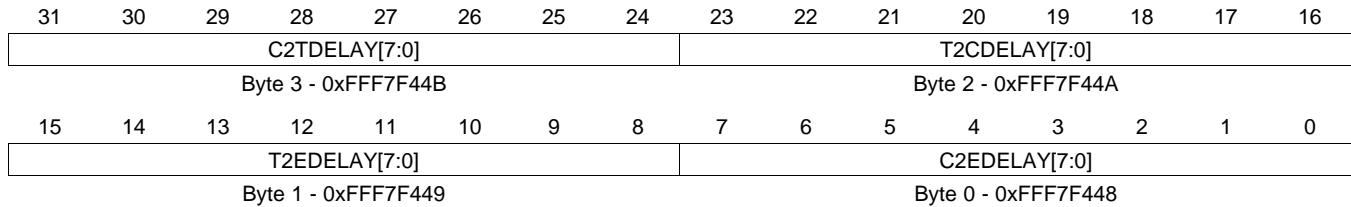


### 1.3 Endianism Considerations

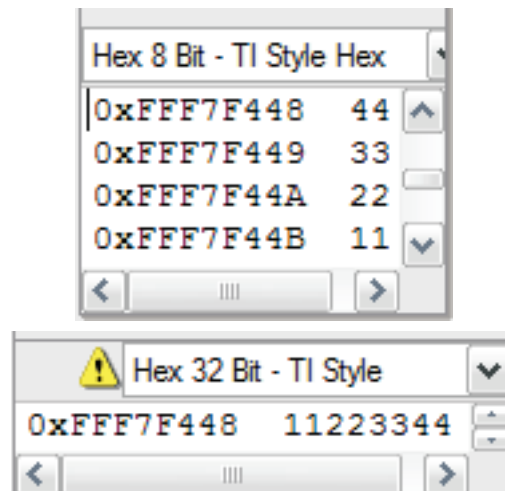
#### 1.3.1 RM48x: Little Endian (LE)

The RM48x family is based on the ARM® Cortex™-R4F core. ARM has designed this core to be used in big-endian and little-endian systems. For the TI RM48x family, the endianness has been configured to little-endian. Little-endian systems store the least-significant byte of a multi-byte data field in the lowest memory address. Also, the address of the multi-byte data field is the lowest address. Below is an example of the physical addresses of individual bytes.

Figure 1-2. Example: SPIDELAY – 0xFFFF7F448



32-bit accesses to this register should use the lowest address i.e. 0xFFFF7F448. Writing 0x11223344 to address 0xFFFF7F448 shows the following when viewing the memory in 8-bit and 32-bit modes.



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

```

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

```





## Architecture

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

Topic	Page
<b>2.1 Introduction .....</b>	<b>92</b>
<b>2.2 Clocks .....</b>	<b>96</b>
<b>2.3 Memory Organization .....</b>	<b>103</b>
<b>2.4 Exceptions .....</b>	<b>115</b>
<b>2.5 System and Peripheral Control Registers .....</b>	<b>118</b>

## 2.1 Introduction

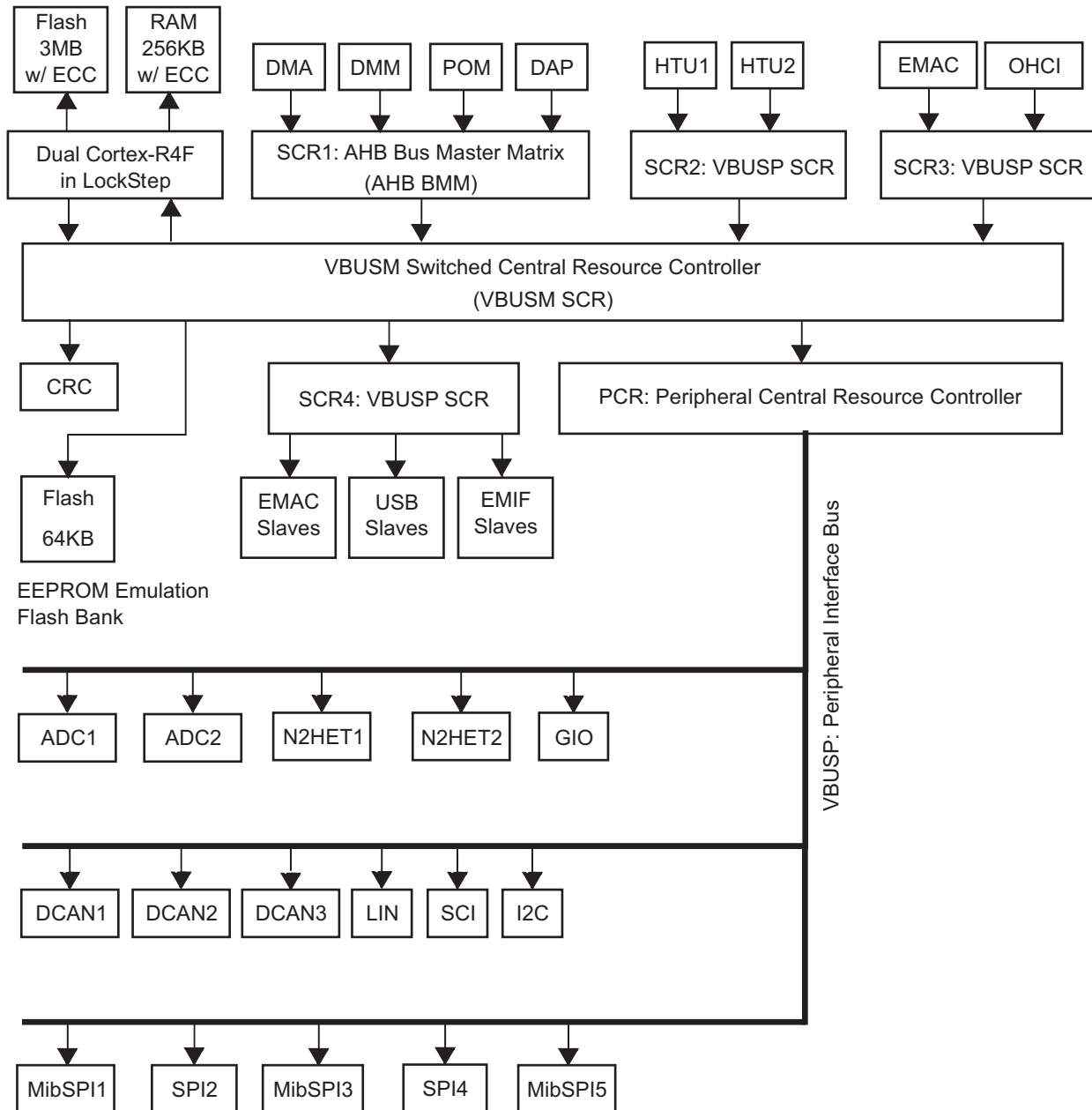
The RM48x family of microcontrollers is based on the Texas Instruments TMS570 Architecture. This chapter describes specific aspects of the architecture as applicable to the RM48x family of microcontrollers.

### 2.1.1 Architecture Block Diagram

The RM48x microcontrollers are based on the TMS570 Platform architecture, which defines the interconnect between the bus masters and the bus slaves.

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

**Figure 2-1. Architectural Block Diagram**



## 2.1.2 Definitions of Terms

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

**Table 2-1. Definition of Terms**

Acronym / Term	Full Form	Description
ECC	Error Correction Code	This is a code that is used by the Single Error Correction Double Error Detection (SECCDED) logic inside the two Cortex-R4F processors (CPUs). There are 8 bits of ECC for every 64 bits of data accessed from the CPU tightly-coupled memories (flash and RAM).
Lockstep	–	This is the mode of operation of the dual ARM Cortex-R4F CPUs. The outputs of the two CPUs are compared on each CPU clock cycle. Any miscompare is flagged as an error of the highest severity level.
DMA	Direct Memory Access	The DMA module is used for transferring 8-, 16-, 32- or 64-bit data across the entire device memory map. The DMA module is one of the bus masters on the device. That is, it can initiate a read or a write transaction.
DMM	Data Modification Module	The DMM allows a tool to use the special DMM I/O interface to modify any data value in any RAM on the device. This modification is done with minimal interruption to the application execution, and can be used for calibration of application algorithms. The DMM is also a bus master in this device.
POM	Parameter Overlay Module	The parameter overlay module redirects accesses to a programmable region in flash memory (read-only) to a RAM memory, either on-chip or via the external memory interface (EMIF). This allows a user to evaluate the impact of changing values of constants stored in the flash memory without actually having to erase and reprogram the flash. The POM is also a bus master in this device.
DAP	Debug Access Port	The DAP allows a tool such as a debugger to read from or write to any region in the device memory map. The DAP is a bus master in this device.
SCR1:AHB BMM	AMBA High-performance Bus Matrix Module	The DMA, DMM, POM and DAP all act as masters on the AMBA High-performance Bus (AHB). The BMM arbitrates between concurrent accesses by these masters using a fixed priority scheme. The modules in descending order of priority are POM → DMA → DMM → DAP.
HTUx	High-end timer Transfer Unit	The HTU is a dedicated transfer unit for the New Enhanced High-End Timer module. The HTU has a native interface to the N2HET RAM, and is used to transfer data to / from the N2HET RAM from / to another region in the device memory map. There is one HTU per N2HET module, so that there are 2 HTU modules on the device. The HTUx are bus masters in this device.
SCR2:VBUSP SCR	VBUSP Switched Central Resource Controller	The SCR2 arbitrates between concurrent accesses by the HTU1 and HTU2. A round-robin priority scheme is used between the HTU1 and HTU2.
EMAC	Ethernet Media Access Controller	The EMAC has a dedicated DMA-type component that is used to transfer data to / from the EMAC descriptor memory from / to another memory in the device memory map. This DMA-type component of the EMAC is a bus master in this device.
SCR3:VBUSP SCR	VBUSP Switched Central Resource Controller	The SCR3 arbitrates between concurrent accesses by the EMAC and another module that is not available in this configuration of the device.
VBUSM SCR	VBUSM Switched Central Resource Controller	This is the main device SCR. It arbitrates between the accesses from multiple bus masters to the bus slaves using a round robin priority scheme.
CRC	Cyclic Redundancy Checker	The CRC module provides two channels to perform background signature verification on any memory region. It also supports maximum-length Parallel Signature Analysis (PDS) based on a 64-bit primitive polynomial. The CRC module is a bus slave in this device.
EEPROM Emulation Flash Bank	Emulated Electrically Erasable Programmable Read-Only Memory	This is a flash bank that is dedicated for use as an emulated EEPROM. This device supports 64KB of flash for emulated EEPROM.
SCR4: VBUSP SCR	VBUSP Switched Central Resource Controller	The SCR4 is used to decode the accesses to the bus slaves for the EMAC and EMIF modules. SCR4 is a bus slave in this device.

**Table 2-1. Definition of Terms (continued)**

Acronym / Term	Full Form	Description
EMAC slaves	Ethernet Media Access Controller slave ports	There are four EMAC slaves: <ol style="list-style-type: none"> <li>1. EMAC Control Module: this provides an interface between the EMAC and MDIO modules and the bus masters. It also includes 8KB of RAM to hold EMAC packet buffer descriptors.</li> <li>2. EMAC: The EMAC module interfaces to the other devices on the Ethernet Network using the Media Independent Interface (MII) or Reduced Media Independent Interface (RMII).</li> <li>3. Management Data Input / Output (MDIO): The MDIO module is used to manage the physical layer (PHY) device connected to the EMAC module.</li> <li>4. Communications Port Programming Interface (CPPI): This is the 8KB of RAM used to hold the EMAC packet buffer descriptors.</li> </ol>
EMIF slaves	External Memory Interface slave ports	There are five EMIF slaves:- External SDRAM memory: EMIF chip select 0- External asynchronous memories: EMIF chip selects 2, 3 and 4- EMIF module control and status registers
PCR	Peripheral Central Resource controller	The PCR manages the accesses to the peripheral registers and peripheral memories. It provides a global reset for all the peripherals. It also supports the capability to selectively enable or disable the clock for each peripheral individually. The PCR also manages the accesses to the system module registers required to configure the device's clocks, interrupts, etc. The system module registers also include status flags for indicating exception conditions – resets, aborts, errors, interrupts.
ADCx	Analog-to-Digital Converter	The ADC uses the Successive Approximation Register architecture. It features a selectable 10-bit or 12-bit resolution. The ADC module also includes a RAM to hold the conversion results. A digital logic wrapper manages accesses to the control and status registers. There are two ADC modules on this device.
N2HETx	New Enhanced High-End Timer	The N2HET is an advanced intelligent timer that provides sophisticated timing functions for real-time applications. The timer is software-controlled, using a reduced instruction set, with a specialized timer micromachine and an attached I/O port. The N2HET can be used for pulse width modulated outputs, capture or compare inputs, or general-purpose I/O.
GIO	General-purpose Input/Output	The GIO module allows up to 16 terminals to be used as general-purpose Input or Output. Each of these are also capable of generating an interrupt to the CPU.
DCANx	Controller Area Network controller	The DCAN supports the CAN 2.0B protocol standard and uses a serial, multi-master communication protocol that efficiently supports distributed real-time control with robust communication rates of up to 1 megabit per second (Mbps). The DCAN is ideal for applications operating in noisy and harsh environments (e.g., automotive and industrial fields) that require reliable serial communication or multiplexed wiring.
LIN	Local Interconnect Network controller	The LIN module supports the Local Interconnect standard revision 2.1 and can be used as a UART in full-duplex mode using the standard Non-Return-to-Zero (NRZ) format.
SCI	Serial Communication Interface	The SCI module supports the standard UART in full-duplex mode using the standard Non-Return-to-Zero (NRZ) format.
I2C	Inter-Integrated Circuit controller	The I2C module is a multi-master communication module providing an interface between the device and an I2C-compatible device via the I2C serial bus. The I2C supports both 100 Kbps and 400 Kbps speeds.
SPIx	Serial Peripheral Interface	The SPIx modules provide a clocked serial communication interface for reliable communication between the device and other serial devices with the standard SPI interface. There are two SPI modules on this device.
MibSPIx	Multi-Buffered Serial Peripheral Interface	The MibSPIx modules also support the standard SPI communication protocol. The transfers are all grouped into transfer chunks called “transfer groups”. These transfer groups are made up of one or more buffers in the MibSPIx RAM. The RAM is used to hold the control information and data to be transmitted, as well as the status information and data that is received. There are three MibSPI modules in this device.

### 2.1.3 Bus Master / Slave Access Privileges

This device implements some restrictions on the bus slave access privileges in order to improve the overall throughput of the interconnect shown in [Figure 2-1](#).

**Table 2-2. Bus Master / Slave Access Privileges**

Bus Masters	Master Id	Bus Slaves Being Accessed				
		Non-CPU Accesses to CPU Flash and RAM	EEPROM Bank, ECC bits, OTP regions	CRC Module	EMAC and EMIF Slaves	PCR Modules
CPU Read	0	Allowed	Allowed	Allowed	Allowed	Allowed
CPU Write	1	Not allowed	Allowed	Allowed	Allowed	Allowed
POM	2	Allowed	Allowed	Allowed	Allowed	Allowed
DMA	3	Allowed	Allowed	Allowed	Allowed	Allowed
DMM	4	Allowed	Allowed	Allowed	Allowed	Allowed
DAP	5	Allowed	Allowed	Allowed	Allowed	Allowed
HTU1	6	Not allowed	Allowed	Allowed	Allowed	Allowed
HTU2	8	Not allowed	Allowed	Allowed	Allowed	Allowed
EMAC	9	Not allowed	Allowed	Not allowed	Allowed	Not allowed

### 2.1.4 Device Endianness

This device is configured to be in the 32-bit word-invariant Big Endian (BE-32) mode. In this format, the processor stores the most significant byte of a word at the lowest-numbered byte, and the least significant byte at the highest-numbered byte. Therefore, byte 0 of the memory system connects to data lines 31-24. [Table 2-3](#) shows this.

**Table 2-3. 32-bit Word-Invariant Big Endian Format**

Bits					Word Address			
31	24	23	16	15		08	07	00
0			1		2		3	0
4			5		6		7	4
8			9		10		11	8

- Most-significant byte is at lowest address
- Word is addressed by byte-address of most significant byte

## 2.2 Clocks

This section describes the clocking structure of the RM48x microcontrollers.

### 2.2.1 Clock Sources

The devices support up to 7 clock sources. These are shown in [Table 2-4](#) below.

**Table 2-4. Clock Sources**

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

The electrical specifications as well as timing requirements for each of the above clock sources are specified in the device datasheet.

#### 2.2.1.1 Enabling / Disabling Clock Sources

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

Each bit in these registers corresponds to the clock source number indicated in [Table 2-4](#). For example, setting bit 1 in the CSDIS or CSDISSET registers disables the PLL#1.

---

**NOTE: Disabling the Main Oscillator or HF LPO**

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

---

The clock source is only disabled once there is no active clock domain that is using that clock source. Also check the "Oscillator and PLL" user guide for more information on enabling / disabling the oscillator and PLL.

On the RM48x microcontrollers, the clock sources 0, 4 and 5 are enabled by default.



### 2.2.1.2 Checking for Valid Clock Sources

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

## 2.2.2 Clock Domains

The clocking on this device is divided into multiple clock domains for flexibility in control as well as clock source selection. There are 10 clock domains on this device. Each of these are described in [Table 2-5](#) below.

**Table 2-5. Clock Domains**

Clock Domain	Clock Domain#	Default Source	SourceSelect ion Register	Special Considerations
GCLK	0	OSCIN	GHVSRC	<ul style="list-style-type: none"> <li>Always the same frequency as HCLK</li> <li>In phase with HCLK</li> <li>Is disabled separately from HCLK via the CDDISx registers bit 0</li> <li>Can be divided by 1 up to 8 when running CPU selftest (LBIST) using the CLKDIV field of the STCCLKDIV register at address 0xFFFFE108</li> </ul>
GCLK2	0	OSCIN	GHVSRC	<ul style="list-style-type: none"> <li>Always the same frequency as GCLK</li> <li>2 cycles delayed from GCLK</li> <li>Is disabled along with GCLK</li> <li>Gets divided by the same divider setting as that for GCLK when running CPU selftest (LBIST)</li> </ul>
HCLK	1	OSCIN	GHVSRC	<ul style="list-style-type: none"> <li>Is disabled via the CDDISx registers bit 1</li> </ul>
VCLK	2	OSCIN	GHVSRC	<ul style="list-style-type: none"> <li>Divided down from HCLK</li> <li>Can be HCLK/1, HCLK/2,... or HCLK/16</li> <li>Is disabled separately from HCLK via the CDDISx registers bit 2</li> </ul>
VCLK2	3	OSCIN	GHVSRC	<ul style="list-style-type: none"> <li>Divided down from HCLK</li> <li>Can be HCLK/1, HCLK/2,... or HCLK/16</li> <li>Frequency must be an integer multiple of VCLK frequency</li> <li>Is disabled separately from HCLK via the CDDISx registers bit 3</li> </ul>
VCLK3	8	OSCIN	GHVSRC	<ul style="list-style-type: none"> <li>Divided down from HCLK</li> <li>Can be HCLK/1, HCLK/2,... or HCLK/16</li> <li>Is disabled separately from HCLK via the CDDISx registers bit 8</li> </ul>
VCLKA1	4	VCLK	VCLKASRC	<ul style="list-style-type: none"> <li>Defaults to VCLK as the source</li> <li>Frequency can be as fast as HCLK frequency</li> <li>Is disabled via the CDDISx registers bit 4</li> </ul>
VCLKA2	5	VCLK	VCLKASRC	<ul style="list-style-type: none"> <li>Defaults to VCLK as the source</li> <li>Frequency can be as fast as HCLK frequency</li> <li>Is disabled via the CDDISx registers bit 5</li> </ul>
VCLKA4	11	VCLK	VCLKACON1	<ul style="list-style-type: none"> <li>Defaults to VCLK as the source</li> <li>Frequency can be as fast as HCLK frequency</li> <li>Is disabled via the CDDISx registers bit 11</li> </ul>
RTICKL	6	VCLK	RCLKSRC	<ul style="list-style-type: none"> <li>Defaults to VCLK as the source</li> <li>If a clock source other than VCLK is selected for RTICKL, then the RTICKL frequency must be less than or equal to VCLK/3</li> <li>Application can ensure this by programming the RTI1DIV field of the RCLKSRC register, if necessary</li> <li>Is disabled via the CDDISx registers bit 6</li> </ul>

Each of the control registers indicated in the above table are defined in [Section 2.5](#). The AC timing characteristics for each clock domain are specified in the device datasheet.

### 2.2.2.1 Enabling / Disabling Clock Domains

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

Each bit in these registers corresponds to the clock domain number indicated in [Table 2-5](#). For example, setting bit 1 in the CDDIS or CDDISSET registers disables the HCLK clock domain. The clock domain will be turned off only when every module that uses the HCLK domain gives the “permission” for HCLK to be turned off.

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

### 2.2.2.2 Mapping Clock Sources to Clock Domains

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

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

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

- **Selecting clock source for VCLKA1 and VCLKA2 domains**

The clock source for VCLKA1 and VCLKA2 domains is selected via the VCLKASRC register. The default source for the VCLKA1 and VCLKA2 domains is the VCLK.

- **Selecting clock source for VCLKA4 domain**

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

- **Selecting clock source for RTICK domain**

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

---

**NOTE: Selecting a clock source for RTICK that is not VCLK**

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

---

### 2.2.3 Low Power Modes

All clock domains are active in the normal operating mode. This is the default mode of operation. As described in [Section 2.2.1.1](#) and [Section 2.2.2.1](#), the application can choose to disable any particular clock source and domain that it does not plan to use. Also, the peripheral central resource controller (PCR) has control registers to enable / disable the peripheral clock (VCLK) for each peripheral select. This offers the application a large number of choices for enabling / disabling clock sources, or clock domains, or clocks to specific peripherals.

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

**Table 2-6. Typical Low-Power Modes**

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

### 2.2.3.1 Typical Software Sequence to Enter a Low-Power Mode

1. Program the flash banks and flash pump fall-back modes to be “sleep”.  
The flash pump transitions from active to sleep mode only after all the flash banks have switched from active to sleep mode. The flash banks start switching from active to sleep mode only after the banks are not accessed for at least a duration defined by the Active Grace Period (AGP) parameter configured for the banks. Refer the flash module user guide for more details.
2. Disable the clock sources that are not required to be kept active.  
A clock source does not get disabled until all clock domains using that clock source are disabled first, or are configured to use an alternate clock source.
3. Disable the clock domains that are not required to be kept active.  
A clock domain does not get disabled until all modules using that clock domain “give their permission” for that clock domain to be turned off.
4. Idle the Cortex-R4F core.  
The ARM Cortex-R4F CPU has internal power management logic, and requires a dedicated instruction to be used in order to enter a low power mode. This is the Wait For Interrupt (WFI) instruction. When a WFI instruction is executed, the Cortex-R4F core flushes its pipeline, flushes all write buffers, and completes all pending bus transactions. At this time the core indicates to the system that the clock to the core can be stopped. This indication is used by the Global Clock Module (GCM) to turn off the CPU clock domain (GCLK) if the CDDIS register bit 0 is set.

### 2.2.3.2 Special Considerations for Entry to Low Power Modes

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

### 2.2.3.3 Selecting Clock Source Upon Wake Up

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

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

### 2.2.4 Clock Test Mode

The RM48x microcontrollers support a test mode which allows a user to bring out several different clock sources and clock domains on to the ECLK terminal. This is very useful information for debug purposes. Each clock source also has a corresponding clock source valid status flag in the Clock Source Valid Status (CSVSTAT) register. The clock source valid status flags can also be brought out on to the NHET1[12] terminal in this clock test mode.

The clock test mode is controlled by the CLKTEST register in the system module register frame.

**Figure 2-2. Clock Test Register (CLKTEST) [offset = FFFF FFF8Ch]**

31	27	26	25	24	23	20	19	16
Reserved	ALTLIMPCLOCK ENABLE	RANGEDET CTRL	RANGEDETENA SSEL	Reserved	CLK_TEST_EN			
R-0	R/W-0	R/W-0	R/W-0	R-0	R/W-Ah			
15	Reserved				4	3	0	
R-0						SEL_ECP_PIN		
R-0						R/W-0		

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

The clock test mode is enabled by writing 0x5 to the CLK\_TEST\_EN field.

The signal to be brought out on to the ECLK terminal is defined by the SEL\_ECP\_PIN field, and the signal to be brought out on to the NHET1[12] terminal is defined by the SEL\_GIO\_PIN field.

The choices for these selections are defined in [Table 2-7](#) below.

**Table 2-7. Clock Test Mode Options**

SEL_ECP_PIN	Signal on ECLK	SEL_GIO_PIN	Signal on NHET1[12]
0000	Oscillator	0000	Oscillator Valid status
0001	PLL1 free-running clock output	0001	PLL1 Valid status
0010	Reserved	0010	Reserved
0011	EXTCLKIN1	0011	Reserved
0100	CLK80K	0100	Reserved
0101	CLK10M	0101	CLK10M Valid status
0110	PLL2 free-running clock output	0110	PLL2 Valid Status
0111	EXTCLKIN2	0111	Reserved
1000	GCLK	1000	CLK80K
1001	RTI Base	1001	Reserved
1010	Reserved	1010	Reserved
1011	AVCLK1	1011	Reserved
1100	AVCLK2	1100	Reserved
1101	Reserved	1101	Reserved
1110	AVCLK4	1110	Reserved
1111	Reserved	1111	Reserved

## 2.2.5 Safety Considerations for Clocks

The RM48x microcontrollers are targeted for use in several safety-critical applications. The following sections describe the internal or external monitoring mechanisms that detect and signal clock source failures.

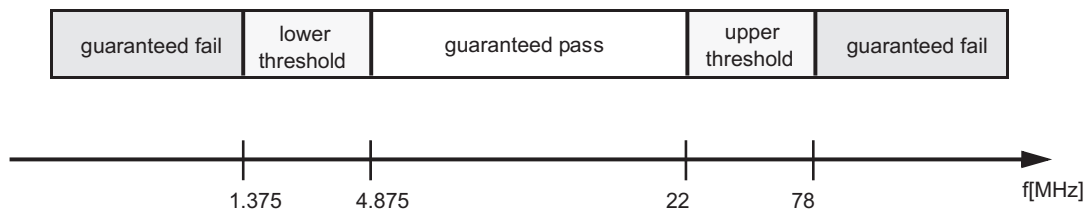
### 2.2.5.1 Oscillator Monitor

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

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

The untrimmed HF LPO frequency on this device can range from 5.5MHz to 19.5MHz. This results in a valid OSCIN frequency range depicted in Figure 2-3 below.

Figure 2-3. LPO and Clock Detection, Untrimmed HF LPO



The application can select the device response to an oscillator fail indication. Please refer to the “Oscillator and PLL” user guide for more details on the oscillator monitoring and the system response choices.

### 2.2.5.2 PLL Slip Detector

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

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

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

### 2.2.5.3 External Clock Monitor

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

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

### 2.2.5.4 Dual-Clock Comparators

The microcontrollers include two instances of the dual-clock comparator module. This module includes two down counters which independently count from two separate seed values at the rate of two independent clock frequencies. One of the clock inputs is a reference clock input, selectable between the main oscillator or the HF LPO in functional mode. The second clock input is selectable from among a set of defined signals as described in Section 2.2.5.4.1 and Section 2.2.5.4.2. This mechanism can be used to use a known-good clock to measure the frequency of another clock.

### 2.2.5.4.1 DCC1

**Table 2-8. DCC1 Counter 0 Clock Inputs**

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

**Table 2-9. DCC1 Counter 1 Clock / Signal Inputs**

Key [3–0]	Clock Source [3–0]	Clock / Signal Name
0xA	0x0	PLL1 free-running output
	0x1	PLL2 free-running output
	0x2	LF LPO
	0x3	HF LPO
	0x4	Reserved
	0x5	EXTCLKIN1
	0x6	EXTCLKIN2
	0x7	Reserved
	0x8 — 0xF	VCLK
others	any value	NHET1[31]

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

### 2.2.5.4.2 DCC2

**Table 2-10. DCC2 Counter 0 Clock Inputs**

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

**Table 2-11. DCC2 Counter 1 Clock / Signal Inputs**

Key [3–0]	Clock Source [3–0]	Clock / Signal Name
0xA	0x0 — 0x7	Reserved
	0x8 — 0xF	VCLK
others	any value	NHET2[0]

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

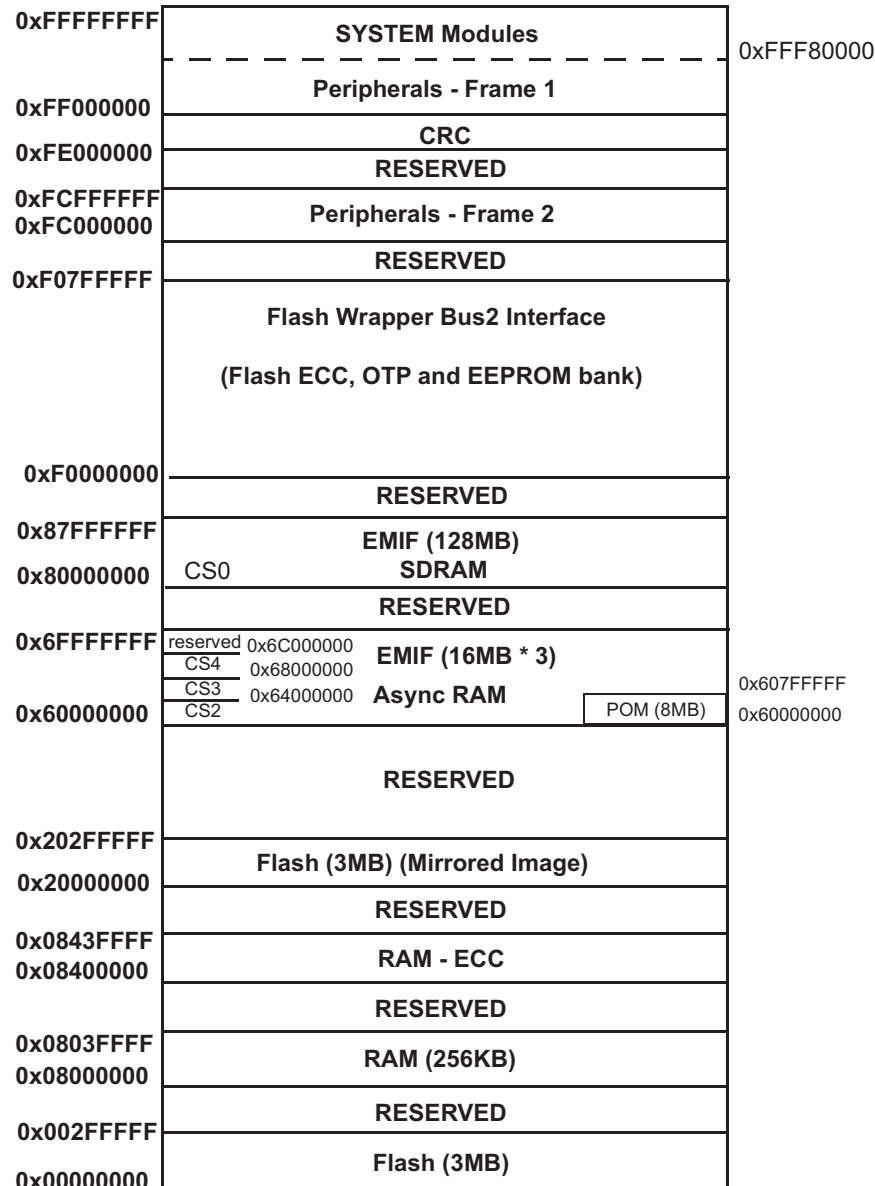


## 2.3 Memory Organization

### 2.3.1 Memory Map Overview

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

Figure 2-4. Memory Map



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

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

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

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

### 2.3.2 Memory Map Table

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

Table 2-12 shows the starting and ending addresses of each module's register frame and any associated memory. The table also shows the response generated by the module or the interconnect whenever an access is made to an unimplemented location inside the register or memory frame.

**Table 2-12. Module Registers / Memories Memory Map**

Name	Memory Select	Frame Address		Frame Size	Actual Size	Response for Access to Unimplemented Location in Frame
		Start	End			
CPU Tightly-Coupled Memories						
TCM Flash	CS0	0x0000_0000	0x00FF_FFFF	16MB	3MB	Abort
TCM RAM + RAM ECC	CSRAM0	0x0800_0000	0x0BFF_FFFF	64MB	256KB	
Mirrored Flash	Flash mirror frame	0x2000_0000	0x20FF_FFFF	16MB	3MB	
External Memory Accesses						
EMIF Chip Select 2 (asynchronous)	EMIF select 2	0x6000_0000	0x63FF_FFFF	64MB	16MB	Access to "Reserved" space will generate Abort
EMIF Chip Select 3 (asynchronous)	EMIF select 3	0x6400_0000	0x67FF_FFFF	64MB	16MB	
EMIF Chip Select 4 (asynchronous)	EMIF select 4	0x6800_0000	0x6BFF_FFFF	64MB	16MB	
EMIF Chip Select 0 (synchronous)	EMIF select 0	0x8000_0000	0x87FF_FFFF	128MB	128MB	
Flash Bus2 Interface: OTP, ECC, EEPROM Bank						
Customer OTP, TCM Flash Bank 0		0xF000_0000	0xF000_1FFF	8KB	4KB	Abort
Customer OTP, TCM Flash Bank 1		0xF000_2000	0xF000_3FFF	8KB	4KB	
Customer OTP, EEPROM Bank		0xF000_E000	0xF000_FFFF	8KB	4KB	
Customer OTP-ECC, TCM Flash Bank 0		0xF004_0000	0xF004_03FF	1KB	512B	
Customer OTP-ECC, TCM Flash Bank 1		0xF004_0400	0xF004_07FF	1KB	512B	
Customer OTP-ECC, EEPROM Bank		0xF004_1C00	0xF004_1FFF	1KB	512KB	
TI OTP, TCM Flash Bank 0		0xF008_0000	0xF008_1FFF	8KB	4KB	
TI OTP, TCM Flash Bank 1		0xF008_2000	0xF008_3FFF	8KB	4KB	
TI OTP, EEPROM Bank		0xF008_E000	0xF008_FFFF	8KB	4KB	
TI OTP-ECC, TCM Flash Bank 0		0xF00C_0000	0xF00C_03FF	1KB	512B	
TI OTP-ECC, TCM Flash Bank 1		0xF00C_0400	0xF00C_07FF	1KB	512B	
TI OTP-ECC, EEPROM Bank		0xF00C_1C00	0xF00C_1FFF	1KB	512KB	
EEPROM Bank-ECC		0xF010_0000	0xF013_FFFF	256KB	8KB	

**Table 2-12. Module Registers / Memories Memory Map (continued)**

Name	Memory Select	Frame Address		Frame Size	Actual Size	Response for Access to Unimplemented Location in Frame
		Start	End			
EEPROM Bank		0xF020_0000	0xF03F_FFFF	2MB	64KB	Abort
Flash Data Space ECC		0xF040_0000	0xF04F_FFFF	1MB	384KB	
<b>EMAC and EMIF Slaves</b>						
CPPI Memory Slave (Ethernet RAM)		0xFC52_0000	0xFC52_1FFF	8KB	8KB	Abort
CPGMAC Slave (Ethernet Slave)		0xFCF7_8000	0xFCF7_87FF	2KB	2KB	No error
CPGMACSS Wrapper (Ethernet Wrapper)		0xFCF7_8800	0xFCF7_88FF	256B	256B	No error
Ethernet MDIO Interface		0xFCF7_8900	0xFCF7_89FF	256B	256B	No error
EMIF Registers		0xFCFF_E800	0xFCFF_E8FF	256B	256B	Abort
<b>Cyclic Redundancy Checker (CRC) Module Register Frame</b>						
CRC	CRC frame	0xFE00_0000	0xFEFF_FFFF	16MB	512B	Accesses above 0x200 generate abort.
<b>Peripheral Memories</b>						
MIBSPI5 RAM	PCS[5]	0xFF0A_0000	0xFF0B_FFFF	128KB	2KB	Abort for accesses above 2KB
MIBSPI3 RAM	PCS[6]	0xFF0C_0000	0xFF0D_FFFF	128KB	2KB	Abort for accesses above 2KB
MIBSPI1 RAM	PCS[7]	0xFF0E_0000	0xFF0F_FFFF	128KB	2KB	Abort for accesses above 2KB
DCAN3 RAM	PCS[13]	0xFF1A_0000	0xFF1B_FFFF	128KB	2KB	Wrap around for accesses to unimplemented address offsets lower than 0x7FF. Abort generated for accesses beyond offset 0x800.
DCAN2 RAM	PCS[14]	0xFF1C_0000	0xFF1D_FFFF	128KB	2KB	Wrap around for accesses to unimplemented address offsets lower than 0x7FF. Abort generated for accesses beyond offset 0x800.
DCAN1 RAM	PCS[15]	0xFF1E_0000	0xFF1F_FFFF	128KB	2KB	Wrap around for accesses to unimplemented address offsets lower than 0x7FF. Abort generated for accesses beyond offset 0x800.
MIBADC2 RAM	PCS[29]	0xFF3A_0000	0xFF3B_FFFF	128KB	8KB	Wrap around for accesses to unimplemented address offsets lower than 0x1FFF. Abort generated for accesses beyond 0x1FFF.
MIBADC1 RAM	PCS[31]	0xFF3E_0000	0xFF3F_FFFF	128KB	8KB	Wrap around for accesses to unimplemented address offsets lower than 0x1FFF. Abort generated for accesses beyond 0x1FFF.

**Table 2-12. Module Registers / Memories Memory Map (continued)**

Name	Memory Select	Frame Address		Frame Size	Actual Size	Response for Access to Unimplemented Location in Frame
		Start	End			
NHET2 RAM	PCS[34]	0xFF44_0000	0xFF45_FFFF	128KB	16KB	Wrap around for accesses to unimplemented address offsets lower than 0x3FFF. Abort generated for accesses beyond 0x3FFF.
NHET1 RAM	PCS[35]	0xFF46_0000	0xFF47_FFFF	128KB	16KB	Wrap around for accesses to unimplemented address offsets lower than 0x3FFF. Abort generated for accesses beyond 0x3FFF.
HET TU2 RAM	PCS[38]	0xFF4C_0000	0xFF4D_FFFF	128KB	1KB	Abort
HET TU1 RAM	PCS[39]	0xFF4E_0000	0xFF4F_FFFF	128KB	1KB	Abort
<b>Debug Components</b>						
CoreSight Debug ROM	CSCS0	0xFFA0_0000	0xFFA0_0FFF	4KB	4KB	Reads return zeros, writes have no effect
Cortex-R4F Debug	CSCS1	0xFFA0_1000	0xFFA0_1FFF	4KB	4KB	Reads return zeros, writes have no effect
ETM-R4	CSCS2	0xFFA0_2000	0xFFA0_2FFF	4KB	4KB	Reads return zeros, writes have no effect
CoreSight TPIU	CSCS3	0xFFA0_3000	0xFFA0_3FFF	4KB	4KB	Reads return zeros, writes have no effect
POM	CSCS4	0xFFA0_4000	0xFFA0_4FFF	4KB	4KB	Abort
<b>Peripheral Control Registers</b>						
HTU1	PS[22]	0xFFF7_A400	0xFFF7_A4FF	256B	256B	Reads return zeros, writes have no effect
HTU2	PS[22]	0xFFF7_A500	0xFFF7_A5FF	256B	256B	Reads return zeros, writes have no effect
NHET1	PS[17]	0xFFF7_B800	0xFFF7_B8FF	256B	256B	Reads return zeros, writes have no effect
NHET2	PS[17]	0xFFF7_B900	0xFFF7_B9FF	256B	256B	Reads return zeros, writes have no effect
GIO	PS[16]	0xFFF7_BC00	0xFFF7_BCFF	256B	256B	Reads return zeros, writes have no effect
MIBADC1	PS[15]	0xFFF7_C000	0xFFF7_C1FF	512B	512B	Reads return zeros, writes have no effect
MIBADC2	PS[15]	0xFFF7_C200	0xFFF7_C3FF	512B	512B	Reads return zeros, writes have no effect
I2C	PS[10]	0xFFF7_D400	0xFFF7_D4FF	256B	256B	Reads return zeros, writes have no effect
DCAN1	PS[8]	0xFFF7_DC00	0xFFF7_DDFF	512B	512B	Reads return zeros, writes have no effect
DCAN2	PS[8]	0xFFF7_DE00	0xFFF7_DFFF	512B	512B	Reads return zeros, writes have no effect
DCAN3	PS[7]	0xFFF7_E000	0xFFF7_E1FF	512B	512B	Reads return zeros, writes have no effect
LIN	PS[6]	0xFFF7_E400	0xFFF7_E4FF	256B	256B	Reads return zeros, writes have no effect
SCI	PS[6]	0xFFF7_E500	0xFFF7_E5FF	256B	256B	Reads return zeros, writes have no effect
MibSPI1	PS[2]	0xFFF7_F400	0xFFF7_F5FF	512B	512B	Reads return zeros, writes have no effect

**Table 2-12. Module Registers / Memories Memory Map (continued)**

Name	Memory Select	Frame Address		Frame Size	Actual Size	Response for Access to Unimplemented Location in Frame
		Start	End			
SPI2	PS[2]	0xFFFF7_F600	0xFFFF7_F7FF	512B	512B	Reads return zeros, writes have no effect
MibSPI3	PS[1]	0xFFFF7_F800	0xFFFF7_F9FF	512B	512B	Reads return zeros, writes have no effect
SPI4	PS[1]	0xFFFF7_FA00	0xFFFF7_FBF	512B	512B	Reads return zeros, writes have no effect
MibSPI5	PS[0]	0xFFFF7_FC00	0xFFFF7_FDF	512B	512B	Reads return zeros, writes have no effect
<b>System Modules Control Registers and Memories</b>						
DMA RAM	PPCS0	0xFFFF8_0000	0xFFFF8_0FFF	4KB	4KB	Abort
VIM RAM	PPCS2	0xFFFF8_2000	0xFFFF8_2FFF	4KB	1KB	Wrap around for accesses to unimplemented address offsets lower than 0x3FF. Abort generated for accesses beyond 0x3FF.
RTP RAM	PPCS3	0xFFFF8_3000	0xFFFF8_3FFF	4KB	4KB	Abort
Flash Wrapper	PPCS7	0xFFFF8_7000	0xFFFF8_7FFF	4KB	4KB	Abort
eFuse Farm Controller	PPCS12	0xFFFF8_C000	0xFFFF8_CFFF	4KB	4KB	Abort
Power Management Module (PMM)	PPSE0	0xFFFFF_0000	0xFFFFF_01FF	512B	512B	Abort
Test Controller (FMTM)	PPSE1	0xFFFFF_0400	0xFFFFF_07FF	1KB	1KB	Reads return zeros, writes have no effect
PCR registers	PPS0	0xFFFFF_E000	0xFFFFF_E0FF	256B	256B	Reads return zeros, writes have no effect
System Module - Frame 2 (see platform architecture specification)	PPS0	0xFFFFF_E100	0xFFFFF_E1FF	256B	256B	Reads return zeros, writes have no effect
PBIST	PPS1	0xFFFFF_E400	0xFFFFF_E5FF	512B	512B	Reads return zeros, writes have no effect
STC	PPS1	0xFFFFF_E600	0xFFFFF_E6FF	256B	256B	Reads return zeros, writes have no effect
IOMM Multiplexing Control Module	PPS2	0xFFFFF_EA00	0xFFFFF_EBFF	512B	512B	Generates address error interrupt, if enabled
DCC1	PPS3	0xFFFFF_EC00	0xFFFFF_ECFF	256B	256B	Reads return zeros, writes have no effect
DMA	PPS4	0xFFFFF_F000	0xFFFFF_F3FF	1KB	1KB	Reads return zeros, writes have no effect
DCC2	PPS5	0xFFFFF_F400	0xFFFFF_F4FF	256B	256B	Reads return zeros, writes have no effect
ESM	PPS5	0xFFFFF_F500	0xFFFFF_F5FF	256B	256B	Reads return zeros, writes have no effect
CCMR4	PPS5	0xFFFFF_F600	0xFFFFF_F6FF	256B	256B	Reads return zeros, writes have no effect
DMM	PPS5	0xFFFFF_F700	0xFFFFF_F7FF	256B	256B	Reads return zeros, writes have no effect
RAM ECC even	PPS6	0xFFFFF_F800	0xFFFFF_F8FF	256B	256B	Reads return zeros, writes have no effect
RAM ECC odd	PPS6	0xFFFFF_F900	0xFFFFF_F9FF	256B	256B	Reads return zeros, writes have no effect
RTP	PPS6	0xFFFFF_FA00	0xFFFFF_FAFF	256B	256B	Reads return zeros, writes have no effect

**Table 2-12. Module Registers / Memories Memory Map (continued)**

Name	Memory Select	Frame Address		Frame Size	Actual Size	Response for Access to Unimplemented Location in Frame
		Start	End			
RTI + DWWD	PPS7	0xFFFF_FC00	0xFFFF_FCFF	256B	256B	Reads return zeros, writes have no effect
VIM Parity	PPS7	0xFFFF_FD00	0xFFFF_FDFF	256B	256B	Reads return zeros, writes have no effect
VIM	PPS7	0xFFFF_FE00	0xFFFF_FEFF	256B	256B	Reads return zeros, writes have no effect
System Module - Frame 1 (see platform architecture specification)	PPS7	0xFFFF_FF00	0xFFFF_FFFF	256B	256B	Reads return zeros, writes have no effect

### 2.3.3 Flash on Microcontrollers

The RM48x microcontrollers support up to 3MB of flash for use as program memory. This is divided into two separate flash banks, each 1.5MB. The microcontrollers also support a separate 64KB flash bank for use as emulated EEPROM.

#### 2.3.3.1 Flash Bank Sectoring Configuration

Each bank is divided into multiple sectors. A flash sector is the smallest region in the flash bank that must be erased. The sectoring configuration of each flash bank is shown in [Table 2-13](#).

**Table 2-13. Flash Memory Banks and Sectors <sup>(1)</sup>**

Sector NO.	SECTOR SIZE	Low Address	High address
Bank 0: 1.5M Bytes			
0	32K Bytes	0x0000_0000	0x0000_7FFF
1	32K Bytes	0x0000_8000	0x0000_FFFF
2	32K Bytes	0x0001_0000	0x0001_7FFF
3	32K Bytes	0x0001_8000	0x0001_FFFF
4	128K Bytes	0x0002_0000	0x0003_FFFF
5	128K Bytes	0x0004_0000	0x0005_FFFF
6	128K Bytes	0x0006_0000	0x0007_FFFF
7	128K Bytes	0x0008_0000	0x0009_FFFF
8	128K Bytes	0x000A_0000	0x000B_FFFF
9	128K Bytes	0x000C_0000	0x000D_FFFF
10	128K Bytes	0x000E_0000	0x000F_FFFF
11	128K Bytes	0x0010_0000	0x0011_FFFF
12	128K Bytes	0x0012_0000	0x0013_FFFF
13	128K Bytes	0x0014_0000	0x0015_FFFF
14	128K Bytes	0x0016_0000	0x0017_FFFF
Bank 1: 1.5M Bytes			
0	128K Bytes	0x0018_0000	0x0019_FFFF
1	128K Bytes	0x001A_0000	0x001B_FFFF
2	128K Bytes	0x001C_0000	0x001D_FFFF
3	128K Bytes	0x001E_0000	0x001F_FFFF

(1)

1. The Flash banks are 144-bit wide bank with ECC support.
2. The flash bank7 can be programmed while executing code from flash bank0 or bank1.
3. Code execution is not allowed from flash bank7.



**Table 2-13. Flash Memory Banks and Sectors <sup>(1)</sup> (continued)**

Sector NO.	SECTOR SIZE	Low Address	High address
4	128K Bytes	0x0020_0000	0x0021_FFFF
5	128K Bytes	0x0022_0000	0x0023_FFFF
6	128K Bytes	0x0024_0000	0x0025_FFFF
7	128K Bytes	0x0026_0000	0x0027_FFFF
8	128K Bytes	0x0028_0000	0x0029_FFFF
9	128K Bytes	0x002A_0000	0x002B_FFFF
10	128K Bytes	0x002C_0000	0x002D_FFFF
11	128K Bytes	0x002E_0000	0x002F_FFFF
Bank 7: 64K Bytes, dedicated for EEPROM emulation			
0	16K Bytes	0xF020_0000	0xF020_3FFF
1	16K Bytes	0xF020_4000	0xF020_7FFF
2	16K Bytes	0xF020_8000	0xF020_BFFF
3	16K Bytes	0xF020_C000	0xF020_FFFF

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

### 2.3.3.2 ECC Protection for Flash Accesses

The RM48x microcontrollers protect all accesses to the on-chip flash memory by dedicated Single-Error-Correction-Double-Error-Detection (SECEDED) logic.

The accesses to the program memory – flash bank 0 and flash bank 1, are protected by SECEDED logic implemented inside the ARM Cortex-R4F CPU. Accesses to the EEPROM emulation flash bank (bank 7) are protected by dedicated SECEDED logic inside the digital interface to the flash banks.

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

The ECC for the flash array is stored in the flash itself, and is mapped to a region starting at 0xF0400000 for the main flash banks 0 and 1, and to a region starting at 0xF0200000 for the EEPROM emulation flash bank 7.

---

**NOTE: ECC Protection Not Enabled By Default**

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

---

**Code Example for Enabling ECC Protection for Main Flash Accesses:**

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

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

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

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

```

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

```

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

### 2.3.4 On-Chip SRAM

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

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

The RM48x microcontrollers are targeted towards safety-critical applications, and it is critical for any failures in the on-chip SRAM modules to be identified before these modules are used for safety-critical functions. These microcontrollers support a Programmable Built-In Self-Test (PBIST) mechanism that is used to test each on-chip SRAM module for faults. The PBIST is usually run on device start-up as it is a destructive test and all contents of the tested SRAM module are overwritten during the test.

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

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

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

Table 2-14 shows the groupings of the various on-chip memories for PBIST. It also lists the memory types and their assigned RAM Group Select (RGS) and Return Data Select (RDS). Refer to the PBIST user guide for more details on the usage of the RGS and RDS information.

**Table 2-14. PBIST Memory Grouping**

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

**Table 2-14. PBIST Memory Grouping (continued)**

Memory	RAM Group#	Memory Type	RGS	RDS
ESRAM6	22	Single-port	19	0/1 .. 4
ETHERNET	23	Dual-port	20	0 .. 6
	24	Dual-port		
	25	Single-port		
ESRAM8	28	Single-port	25	0/1 .. 4

**Table 2-15** maps the different algorithms supported in application mode for the RAM groups. The table also lists the background pattern options available for each algorithm.

**Table 2-15. PBIST Algorithm Mapping**

No.	ALGO MASK Register Value	Algorithm	Memories Tested	Available Background Patterns	Valid RAM Groups	Valid RAMINFO Register Value
1	0x00000001	triple_read_slow_read	ROM		1,2	0x00000003
2	0x00000002	triple_read_fast_read	ROM		1,2	0x00000003
3	0x00000004	march13n	two_port	0x00000000, 0x96699669, 0x0F0F0F0F, 0xAA55AA55, 0xC3C3C3C3	3,4,5,7,8,9,10,11,12,13,14,15,16,18,19,20,23, 24	0x02CEFFDC
4	0x00000008	march13n	single_port	0x00000000, 0x96699669, 0x0F0F0F0F, 0xAA55AA55, 0xC3C3C3C3	6,17,21,22,25, 28	0x0D310020
5	0x00000010	down1A_red	two_port	0xFFFFFFFF, 0AAAAAAAA	3,4,5,7,8,9,10,11,12,13,14,15,16,18,19,20,23, 24	0x02CEFFDC
6	0x00000020	down1A_red	single_port	0xFFFFFFFF, 0AAAAAAAA	6,17,21,22,25, 28	0x0D310020
7	0x00000040	mapcolumn	two_port	0xFFFFFFFF, 0x00000000	3,4,5,7,8,9,10,11,12,13,14,15,16,18,19,20,23, 24	0x02CEFFDC
8	0x00000080	mapcolumn	single_port	0xFFFFFFFF, 0x00000000	6,17,21,22,25, 28	0x0D310020
9	0x00000100	precharge	two_port	0xFFFFFFFF, 0x00000000	3,4,5,7,8,9,10,11,12,13,14,15,16,18,19,20,23, 24	0x02CEFFDC
10	0x00000200	precharge	single_port	0xFFFFFFFF, 0x00000000	6,17,21,22,25, 28	0x0D310020
11	0x00000400	dtxn2	two_port	0xFFFFFFFF, 0x00000000	3,4,5,7,8,9,10,11,12,13,14,15,16,18,19,20,23, 24	0x02CEFFDC
12	0x00000800	dtxn2	single_port	0xFFFFFFFF, 0x00000000	6,17,21,22,25, 28	0x0D310020
13	0x00001000	pmos_open	two_port	0xFFFFFFFF, 0x00000000	3,4,5,7,8,9,11,14,15,16,18,20, 24	0x028AE5DC
14	0x00002000	pmos_open	single_port	0xFFFFFFFF, 0x00000000	6,17,21,22,25, 28	0x0D310020
15	0x00004000	pmos_open_slice1	two_port	0xFFFFFFFF, 0x00000000	10,12,13,19,23	0x00441A00
16	0x00008000	pmos_open_slice2	two_port	0xFFFFFFFF, 0x00000000	10,12,13,19,23	0x00441A00
17	0x00010000	flip10	two_port	0xFFFFFFFF	3,4,5,7,8,9,10,11,12,13,14,15,16,18,19,20,23, 24	0x02CEFFDC
18	0x00020000	flip10	single_port	0xFFFFFFFF	6,17,21,22,25, 28	0x0D310020
19	0x00040000	iddq	two_port	0x00000000	3,4,5,7,8,9,10,11,12,13,14,15,16,18,19,20,23, 24	0x02CEFFDC
20	0x00080000	iddq	single_port	0x00000000	6,17,21,22,25, 28	0x0D310020

**Table 2-15. PBIST Algorithm Mapping (continued)**

No.	ALGO MASK Register Value	Algorithm	Memories Tested	Available Background Patterns	Valid RAM Groups	Valid RAMINFO Register Value
21	0x00100000	retention	two_port	0x00000000	3,4,5,7,8,9,10,11,12,13,14,15,16,18,19,20,23, 24	0x02CEFFDC
22	0x00200000	retention	single_port	0x00000000	6,17,21,22,25, 28	0x0D310020
23	0x00400000	iddq	two_port	0xFFFFFFFF	3,4,5,7,8,9,10,11,12,13,14,15,16,18,19,20,23, 24	0x02CEFFDC
24	0x00800000	iddq	single_port	0xFFFFFFFF	6,17,21,22,25, 28	0x0D310020
25	0x01000000	retention	two_port	0xFFFFFFFF	3,4,5,7,8,9,10,11,12,13,14,15,16,18,19,20,23, 24	0x02CEFFDC
26	0x02000000	retention	single_port	0xFFFFFFFF	6,17,21,22,25, 28	0x0D310020
27	0x04000000	iddqrowstripe	two_port	0x00000000	3,4,5,7,8,9,10,11,12,13,14,15,16,18,19,20,23, 24	0x02CEFFDC
28	0x08000000	iddqrowstripe	single_port	0x00000000	6,17,21,22,25, 28	0x0D310020
29	0x10000000	iddqrowstripe	two_port	0xFFFFFFFF	3,4,5,7,8,9,10,11,12,13,14,15,16,18,19,20,23, 24	0x02CEFFDC
30	0x20000000	iddqrowstripe	single_port	0xFFFFFFFF	6,17,21,22,25, 28	0x0D310020
31	0x40000000	powerup_invpowerup	two_port	0xAAAAAAAA	33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51	0x0007FFFF
32	0x80000000	powerup_invpowerup	single_port	0xAAAAAAAA	52,53,54,55,56,57,58	0x03F80000

**NOTE: Recommended Memory Test Algorithm**

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

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

**2.3.4.2 Auto-Initialization of On-Chip SRAM Modules**

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

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

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

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

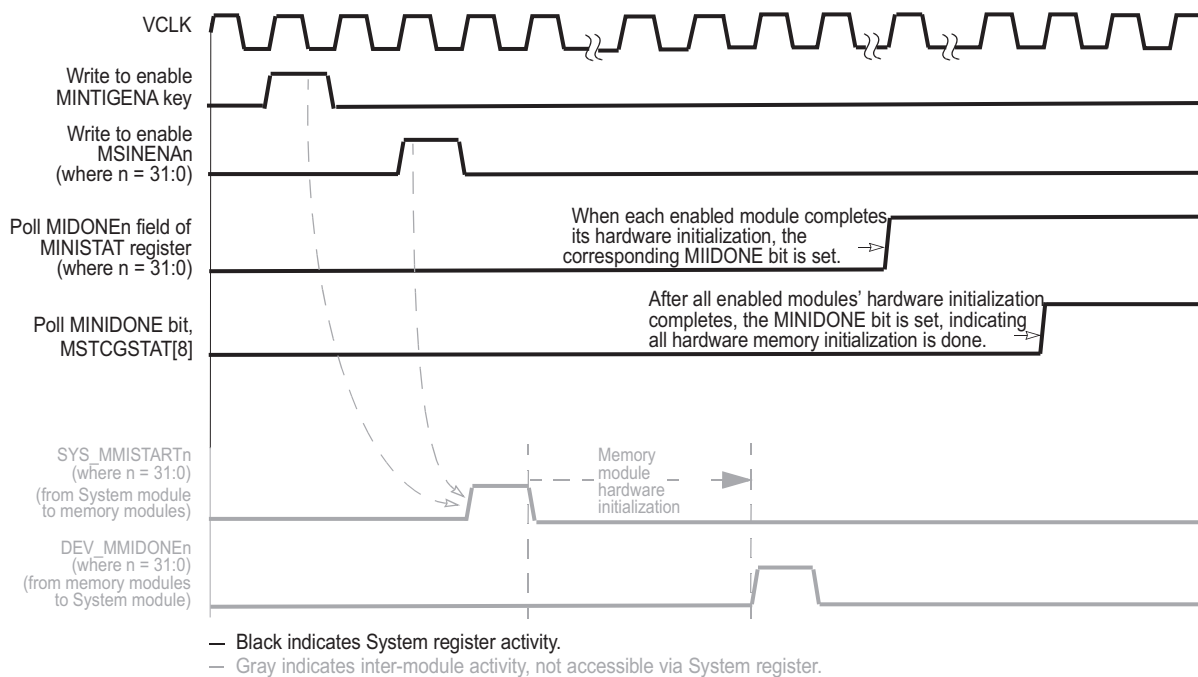
**Auto-Initialization Sequence:**

1. Enable the global hardware memory initialization key by programming 0xA into MINITGCR[3:0], the Memory Initialization Key field (MINITGENA) of the Memory Hardware Initialization Global Control Register (MINITGCR) register.
2. Select the module on which the memory hardware initialization has to be performed by programming the appropriate value into the MSINENA(31–0) bits in the MSINENA register. See [Table 2-16](#).
3. If the global auto-initialization scheme is enabled, the corresponding module will initialize its memories based on its memory error checking scheme (even parity or odd parity or ECC).
4. When the memory initialization is complete, the module will signal “memory initialization done”, which sets the corresponding bit in the system module MIDONE field of the MINISTAT register to indicate the

completion of its memory initialization.

- When the memory hardware initialization completes for all modules, (indicated by each module's MIDONE bit being set), the memory hardware initialization done bit (MINIDONE) is set in the MSTCGSTAT register.

**Figure 2-5. Hardware Memory Initialization Protocol**



**Table 2-16. Memory Initialization Select Mapping <sup>(1)(2)(3)</sup>**

Memory	Address Range		MSINENA Register Bit #
	Start	End	
RAM	0x08000000	0x08013FFF	0
RAM (always ON domain, PD#1)	0x08000000	0x0800FFFF	0
RAM (RAM_PD#1)	0x08010000	0x0801FFFF	0
RAM (RAM_PD#2)	0x08020000	0x0802FFFF	0
RAM (RAM_PD#3)	0x08030000	0x0803FFFF	0
MIBSPI5 RAM	0xFF0A0000	0xFF0BFFFF	12 <sup>(4)</sup>
MIBSPI3 RAM	0xFF0C0000	0xFF0DFFFF	11 <sup>(4)</sup>
MIBSPI1 RAM	0xFF0E0000	0xFF0FFFFFFF	7 <sup>(4)</sup>
DCAN3 RAM	0xFF1A0000	0xFF1BFFFF	10
DCAN2 RAM	0xFF1C0000	0xFF1DFFFF	6
DCAN1 RAM	0xFF1E0000	0xFF1FFFFFFF	5
MIBADC2 RAM	0xFF3A0000	0xFF3BFFFF	14
MIBADC1 RAM	0xFF3E0000	0xFF3FFFFFFF	8
NHET2 RAM	0xFF440000	0xFF57FFFF	15

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

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

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

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

**Table 2-16. Memory Initialization Select Mapping <sup>(1)(2)(3)</sup> (continued)**

Memory	Address Range		MSINENA Register Bit #
	Start	End	
NHET1 RAM	0xFF460000	0xFF47FFFF	3
HET TU2 RAM	0xFF4C0000	0xFF4DFFFF	16
HET TU1 RAM	0xFF4E0000	0xFF4FFFFFFF	4
DMA RAM	0xFFF80000	0xFFF80FFF	1
VIM RAM	0xFFF82000	0xFFF82FFF	2



## 2.4 Exceptions

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

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

Please refer to the ARM Cortex-R4F Technical Reference Manual for complete details on all exceptions.

### 2.4.1 Resets

The RM48x microcontroller can be reset by either of the conditions described in [Table 2-17](#). Each reset condition is indicated in the System Exception Status Register (SYSESR).

**Table 2-17. Causes of Resets**

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

The device nRST terminal is an I/O. It can be driven low by an external circuit to force a warm reset on the microcontroller. This terminal will be driven low as an output for a minimum of 8 peripheral clock (VCLK) cycles for any device system reset condition. As a result the EXT RST bit in the SYSESR register, SYSESR[3], gets set for all reset conditions listed in [Table 2-17](#). The nRST is driven low as an output for a longer duration during device power-up or whenever the power-on reset (nPORRST) is driven low externally. Refer the device datasheet for the electrical and timing specifications for the nRST.

### 2.4.2 Aborts

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

### 2.4.2.1 Prefetch Aborts

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

All prefetch aborts are precise aborts.

### 2.4.2.2 Data Aborts

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

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

### 2.4.2.3 Precise Aborts

A precise abort, also known as a synchronous abort, is one for which the exception is guaranteed to be taken on the instruction that generated the aborting memory access. The abort handler can use the value in the Link Register (r14\_abt) to determine which instruction generated the abort, and the value in the Saved Program Status Register (SPSR\_abt) to determine the state of the processor when the abort occurred.

### 2.4.2.4 Imprecise Aborts

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

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

The RM48x microcontroller architecture applies techniques at the system level to mitigate the impact of imprecise aborts. System level adoption of write status sidebands to the data path allow bus masters to comprehend imprecise aborts, turning them into precise aborts. In cases where this approach is not feasible, buffering bridges or other sources of imprecision may build a FIFO of current transactions such that an imprecise abort may be registered at the point of imprecision for later analysis.

#### **Masking Of Imprecise Aborts:**

The nature of imprecise aborts means that they can occur while the processor is handling a different abort. If an imprecise abort generates a new exception in such a situation, the banked link register (R14\_abt) and the Saved Processor Status Register (SPSR\_abt) values are overwritten. If this occurs before the data is pushed to the stack in memory, the state information about the first abort is lost. To prevent this from happening, the Current Processor Status Register (CPSR) contains a mask bit to indicate that an imprecise abort cannot be accepted, the A-bit. When the A-bit is set, any imprecise abort that occurs is held pending by the processor until the A-bit is cleared, when the exception is actually taken. The A-bit is automatically set when abort, IRQ or FIQ exceptions are taken, and on reset. The application must only clear the A-bit in an abort handler after the state information has either been stacked to memory, or is no longer required.

---

#### **NOTE: Default Behavior for Imprecise Aborts**

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

---

### 2.4.2.5 Conditions That Generate Aborts

An Abort is generated under the following conditions on the RM48x microcontrollers.

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

#### **Illegal Addresses:**

The illegal addresses and the responses to an access to these addresses are defined in [Table 2-12](#).

#### **Addresses Protected By MPU:**

Memory access permissions can be configured via the ARM Cortex-R4F processor's Memory Protection Unit (MPU). Please refer to the ARM Cortex-R4F TRM for more details on the MPU configuration.

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

#### **Protection of Peripheral Register and Memory Frames:**

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

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

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

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

---

**NOTE: No Access Protection for Reads**

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

---

### 2.4.3 System Software Interrupts

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

## 2.5 System and Peripheral Control Registers

The following sections describe the system and peripheral control registers of the RM48x microcontroller.

### 2.5.1 Primary System Control Registers (SYS)

This section describes the SYSTEM registers. These registers are broken up into two separate frames. The start address of the primary system module frame is FFFF FF00h and the end address is FFFF FFFFh. The start address of the secondary system module frame is FFFF E100h and the end address is FFFF E1FFh. The registers support 32-, 16-, and 8-bit writes. The offset is relative to the system module frame start address.

Table 2-18 contains a summary figure of the system control registers.

**Table 2-18. Primary System Control Registers Summary**

Offset	Acronym	Register Description	Section
00	SYSPC1	SYS Pin Control Register 1	<a href="#">Section 2.5.1.1</a>
04h	SYSPC2	SYS Pin Control Register 2	<a href="#">Section 2.5.1.2</a>
08h	SYSPC3	SYS Pin Control Register 3	<a href="#">Section 2.5.1.3</a>
0Ch	SYSPC4	SYS Pin Control Register 4	<a href="#">Section 2.5.1.4</a>
10h	SYSPC5	SYS Pin Control Register 5	<a href="#">Section 2.5.1.5</a>
14h	SYSPC6	SYS Pin Control Register 6	<a href="#">Section 2.5.1.6</a>
18h	SYSPC7	SYS Pin Control Register 7	<a href="#">Section 2.5.1.7</a>
1Ch	SYSPC8	SYS Pin Control Register 8	<a href="#">Section 2.5.1.8</a>
20h	SYSPC9	SYS Pin Control Register 9	<a href="#">Section 2.5.1.9</a>
30h	CSDIS	Clock Source Disable Register	<a href="#">Section 2.5.1.10</a>
34h	CSDISSET	Clock Source Disable Set Register	<a href="#">Section 2.5.1.11</a>
38h	CSDISCLR	Clock Source Disable Clear Register	<a href="#">Section 2.5.1.12</a>
3Ch	CDDIS	Clock Domain Disable Register	<a href="#">Section 2.5.1.13</a>
40h	CDDISSET	Clock Domain Disable Set Register	<a href="#">Section 2.5.1.14</a>
44h	CDDISCLR	Clock Domain Disable Clear Register	<a href="#">Section 2.5.1.15</a>
48h	GHVSR	GCLK, HCLK, VCLK, and VCLK2 Source Register	<a href="#">Section 2.5.1.16</a>
4Ch	VCLKASRC	Peripheral Asynchronous Clock Source Register	<a href="#">Section 2.5.1.17</a>
50h	RCLKSRC	RTI Clock Source Register	<a href="#">Section 2.5.1.18</a>
54h	CSVSTAT	Clock Source Valid Status Register	<a href="#">Section 2.5.1.19</a>
58h	MSTGCR	Memory Self-Test Global Control Register	<a href="#">Section 2.5.1.20</a>
5Ch	MINITGCR	Memory Hardware Initialization Global Control Register	<a href="#">Section 2.5.1.21</a>
60h	MSIENA	Memory Self-Test/Initialization Enable Register	<a href="#">Section 2.5.1.22</a>
64h	MSTFAIL	Memory Self-Test Fail Status Register	<a href="#">Section 2.5.1.23</a>
68h	MSTCGSTAT	MSTC Global Status Register	<a href="#">Section 2.5.1.24</a>
6Ch	MINISTAT	Memory Hardware Initialization Status Register	<a href="#">Section 2.5.1.25</a>
70h	PLLCTL1	PLL Control Register 1	<a href="#">Section 2.5.1.26</a>
74h	PLLCTL2	PLL Control Register 2	<a href="#">Section 2.5.1.27</a>
78h	SYSPC10	SYS Pin Control Register 10	<a href="#">Section 2.5.1.28</a>
7Ch	DIEIDL	Die Identification Register, Lower Word	<a href="#">Section 2.5.1.29</a>
80h	DIEIDH	Die Identification Register, Upper Word	<a href="#">Section 2.5.1.30</a>
88h	LPOMONCTL	LPO/Clock Monitor Control Register	<a href="#">Section 2.5.1.31</a>
8Ch	CLKTEST	Clock Test Register	<a href="#">Section 2.5.1.32</a>
90h	DFTCTRLREG	DFT Control Register	<a href="#">Section 2.5.1.33</a>
94h	DFTCTRLREG2	DFT Control Register 2	<a href="#">Section 2.5.1.34</a>
A0h	GPREG1	General Purpose Register	<a href="#">Section 2.5.1.35</a>
A8h	IMPFAS	Imprecise Fault Status Register	<a href="#">Section 2.5.1.36</a>
ACh	IMPFTADD	Imprecise Fault Write Address Register	<a href="#">Section 2.5.1.37</a>

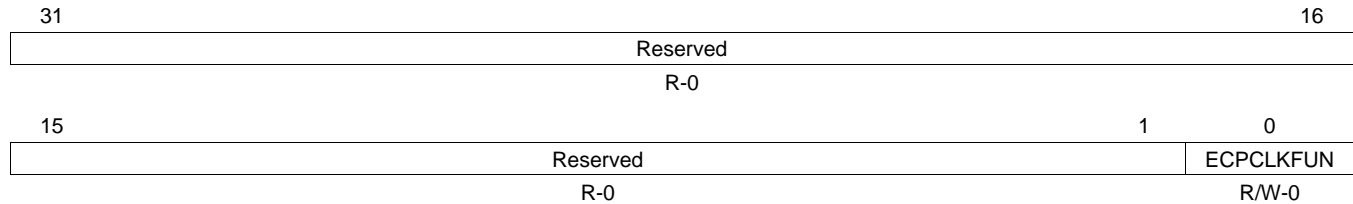
**Table 2-18. Primary System Control Registers Summary (continued)**

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

### 2.5.1.1 SYS Pin Control Register 1 (SYSPC1)

The SYSPC1 register, shown in [Figure 2-6](#) and described in [Table 2-19](#), controls the function of the ECLK pin.

**Figure 2-6. SYS Pin Control Register 1 (SYSPC1) [offset = 00]**



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

**Table 2-19. SYS Pin Control Register 1 (SYSPC1) Field Descriptions**

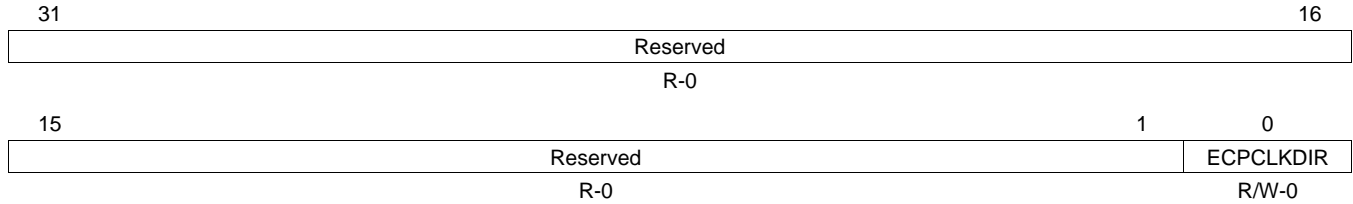
Bit	Field	Value	Description
31-1	Reserved	0	Read returns 0. Writes have no effect.
0	ECPCLKFUN	0 1	ECLK function. This bit changes the function of the ECLK pin. ECLK is in GIO mode. ECLK is in functional mode as a clock output. <b>Note: Proper ECLK duty cycle is not guaranteed until 1 ECLK cycle has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



**2.5.1.2 SYS Pin Control Register 2 (SYSPC2)**

The SYSPC2 register, shown in [Figure 2-7](#) and described in [Table 2-20](#), controls whether the pin is an input or an output when in GIO mode.

**Figure 2-7. SYS Pin Control Register 2 (SYSPC2) [offset = 04h]**



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

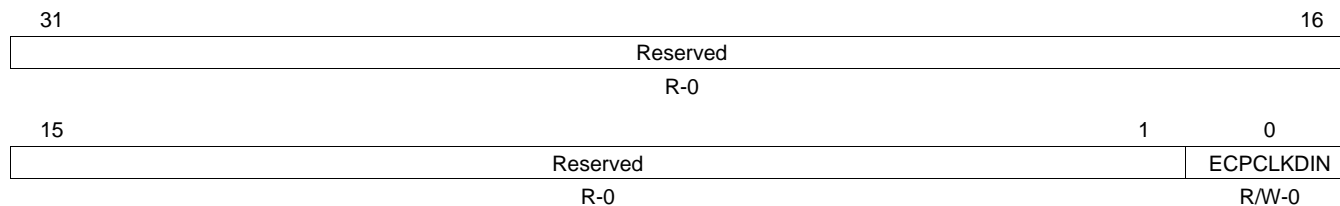
**Table 2-20. SYS Pin Control Register 2 (SYSPC2) Field Descriptions**

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

### 2.5.1.3 SYS Pin Control Register 3 (SYSPC3)

The SYSPC3 register, shown in [Figure 2-8](#) and described in [Table 2-21](#), displays the logic state of the ECLK pin when it is in GIO mode.

**Figure 2-8. SYS Pin Control Register 3 (SYSPC3) [offset = 08h]**



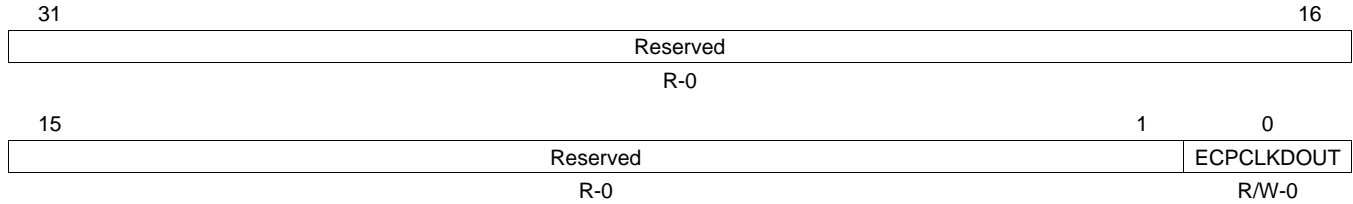
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 2-21. SYS Pin Control Register 3 (SYSPC3) Field Descriptions**

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

**2.5.1.4 SYS Pin Control Register 4 (SYSPC4)**

The SYSPC4 register, shown in [Figure 2-9](#) and described in [Table 2-22](#), controls the logic level output function of the ECLK pin when when it is configured as an output in GIO mode.

**Figure 2-9. SYS Pin Control Register 4 (SYSPC4) [offset = 0Ch]**


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

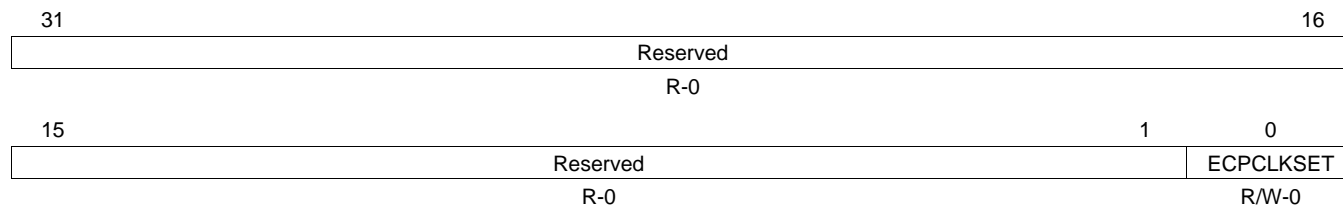
**Table 2-22. SYS Pin Control Register 4 (SYSPC4) Field Descriptions**

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

### 2.5.1.5 SYS Pin Control Register 5 (SYSPC5)

The SYSPC5 register, shown in [Figure 2-10](#) and described in [Table 2-23](#), controls the set function of the ECLK pin when it is configured as an output in GIO mode.

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



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

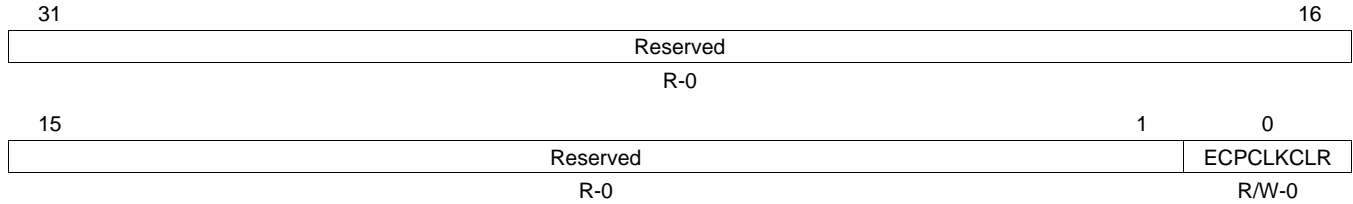
**Table 2-23. SYS Pin Control Register 5 (SYSPC5) Field Descriptions**

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

### 2.5.1.6 SYS Pin Control Register 6 (SYSPC6)

The SYSPC6 register, shown in [Figure 2-11](#) and described in [Table 2-24](#), controls the clear function of the ECLK pin when it is configured as an output in GIO mode..

**Figure 2-11. SYS Pin Control Register 6 (SYSPC6) [offset = 14h]**



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

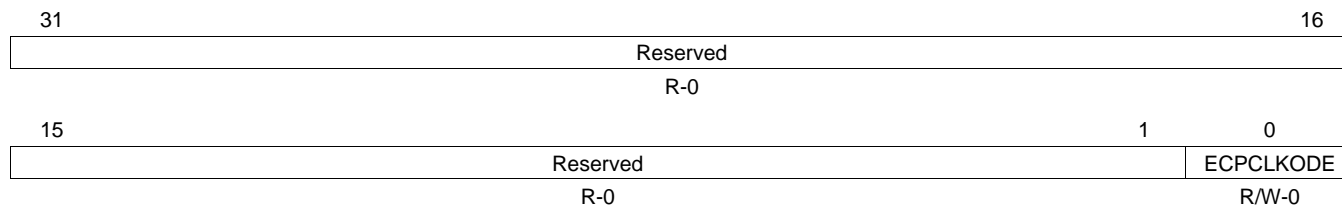
**Table 2-24. SYS Pin Control Register 6 (SYSPC6) Field Descriptions**

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

### 2.5.1.7 SYS Pin Control Register 7 (SYSPC7)

The SYSPC7 register, shown in [Figure 2-12](#) and described in [Table 2-25](#), controls the open drain function of the ECLK pin.

**Figure 2-12. SYS Pin Control Register 7 (SYSPC7) [offset = 18h]**



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

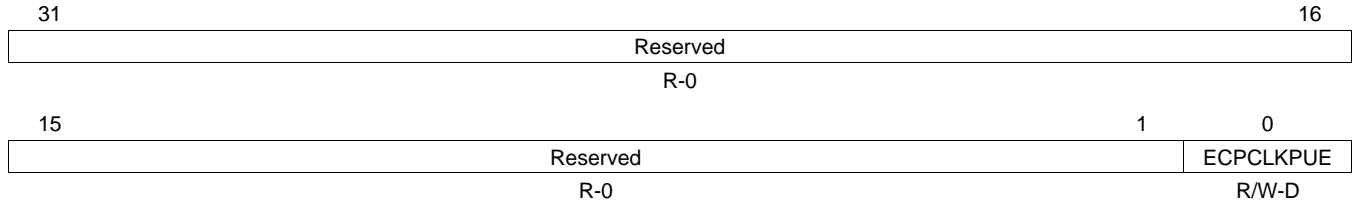
**Table 2-25. SYS Pin Control Register 7 (SYSPC7) Field Descriptions**

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



**2.5.1.8 SYS Pin Control Register 8 (SYSPC8)**

The SYSPC8 register, shown in [Figure 2-13](#) and described in [Table 2-26](#), controls the pull enable function of the ECLK pin when it is configured as an input in GIO mode.

**Figure 2-13. SYS Pin Control Register 8 (SYSPC8) [offset = 1Ch]**


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

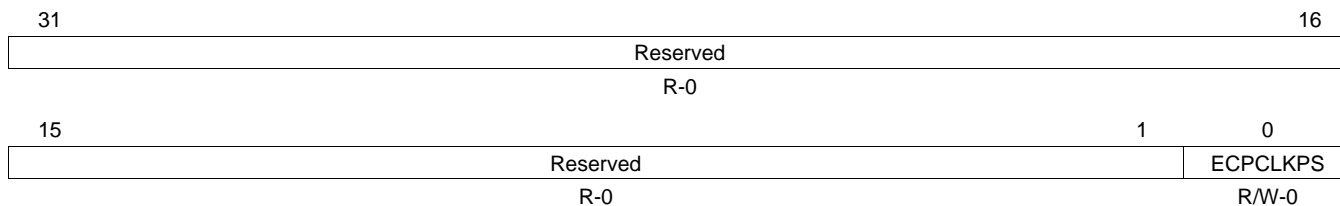
**Table 2-26. SYS Pin Control Register 8 (SYSPC8) Field Descriptions**

Bit	Field	Value	Description
31-1	Reserved	0	Read returns 0. Writes have no effect.
0	ECPCCLKPUE	0 1	ECLK pull enable. Writes to this bit will only take effect when the ECLK pin is configured as an input in GIO mode.  0      ECLK pull enable is active. 1      ECLK pull enable is inactive.  <b>Note: The pull direction (up/down) is selected by the ECPCCLKPS bit in the SYSPC9 register.</b>  <b>Note: The ECLK pin is placed into GIO mode by setting the ECPCCLKFUN bit to 0 in the SYSPC1 register. The ECLK pin is placed in input mode by setting the ECPCCLKDIR bit to 0 in the SYSPC2 register.</b>

### 2.5.1.9 SYS Pin Control Register 9 (SYSPC9)

The SYSPC9 register, shown in [Figure 2-14](#) and described in [Table 2-27](#), controls the pull up/pull down configuration of the ECLK pin when it is configured as an input in GIO mode.

**Figure 2-14. SYS Pin Control Register 9 (SYSPC9) [offset = 20h]**



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

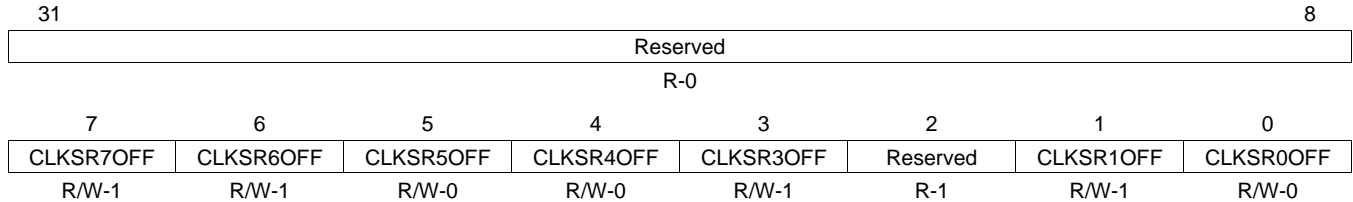
**Table 2-27. SYS Pin Control Register 9 (SYSPC9) Field Descriptions**

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

### 2.5.1.10 Clock Source Disable Register (CSDIS)

The CSDIS register, shown in Figure 2-15 and described in Table 2-28, controls and displays the state of the device clock sources.

**Figure 2-15. Clock Source Disable Register (CSDIS) [offset = 30h]**



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

**Table 2-28. Clock Source Disable Register (CSDIS) Field Descriptions**

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

**Table 2-29. Clock Sources Table**

Clock Source #	Clock Source Name
Clock Source 0	Oscillator
Clock Source1	PLL1
Clock Source 2	Not Implemented
Clock Source 3	EXTCLKIN
Clock Source 4	Low Frequency LPO (Low Power Oscillator) clock
Clock Source 5	High frequency LPO (Low Power Oscillator) clock
Clock Source 6	PLL2
Clock Source 7	EXTCLKIN2

---

**NOTE:** Non implemented clock sources should not be enabled or used.

---

### 2.5.1.11 Clock Source Disable Set Register (CSDISSET)

The CSDISSET register, shown in [Figure 2-16](#) and described in [Table 2-30](#), sets clock sources to the disabled state.

**Figure 2-16. Clock Source Disable Set Register (CSDISSET) [offset = 34h]**

Reserved								31	8
R-0									
7	6	5	4	3	2	1	0		
SET CLKSR7OFF	SET CLKSR6OFF	SET CLKSR5OFF	SET CLKSR4OFF	SET CLKSR3OFF	Reserved	SET CLKSR1OFF	SET CLKSR0OFF		
R/W-1	R/W-1	R/W-0	R/W-0	R/W-1	R-1	R/W-1	R/W-0		

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

**Table 2-30. Clock Source Disable Set Register (CSDISSET) Field Descriptions**

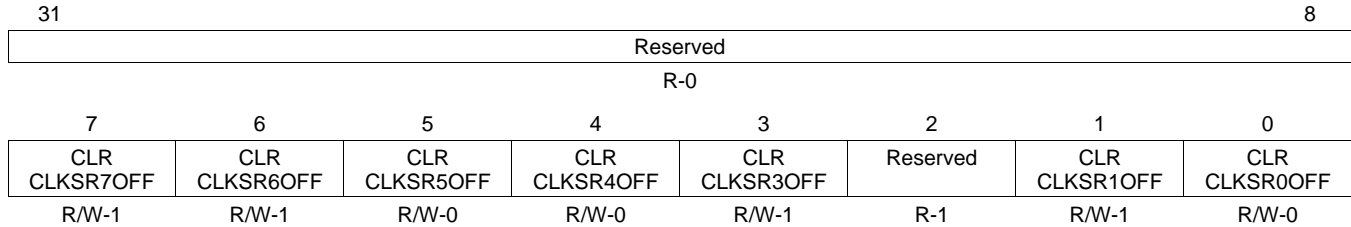
Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-3	SETCLKSR[7-3] OFF	0	Set clock source[7-3] to the disabled state. <i>Read:</i> Clock source[7-3] is enabled. <i>Write:</i> Clock source[7-3] is unchanged.
		1	<i>Read:</i> Clock source[7-3] is disabled. <i>Write:</i>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

**NOTE:** A list of the available clock sources is shown in the [Clock Sources Table](#).

### 2.5.1.12 Clock Source Disable Clear Register (CSDISCLR)

The CSDISCLR register, shown in [Figure 2-17](#) and described in [Table 2-31](#), clears clock sources to the enabled state.

**Figure 2-17. Clock Source Disable Clear Register (CSDISCLR) [offset = 38h]**



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

**Table 2-31. Clock Source Disable Clear Register (CSDISCLR) Field Descriptions**

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

**NOTE:** A list of the available clock sources is shown in the [Clock Sources Table](#).

### 2.5.1.13 Clock Domain Disable Register (CDDIS)

The CDDIS register, shown in Figure 2-18 and described in Table 2-32, controls the state of the clock domains.

**NOTE:** All the clock domains are enabled on wakeup.

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

**Figure 2-18. Clock Domain Disable Register (CDDIS) [offset = 3Ch]**

31	Reserved								16	
R-0										
15	Reserved				12	11	10	9	8	
R-0					R/W-0	R/W-0	R/W-0	R/W-0		
7	6	5	4	3	2	1	0			
Reserved	RTICK1OFF	VCLKA2OFF	VCLKA1OFF	VCLK2OFF	VCLKPOFF	HCLKOFF	GCLKOFF			
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0			

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

**Table 2-32. Clock Domain Disable Register (CDDIS) Field Descriptions**

Bit	Field	Value	Description
31-12	Reserved	0	Read returns 0. Writes have no effect.
11-10	VCLKA[4-3]OFF	0	VCLKA[4-3] domain off. The VCLKA[4-3] domain is enabled.
		1	The VCLKA[4-3] domain is disabled.
9	Reserved		Reads return zero or one and privilege mode writes allowed.
8	VCLK3OFF	0	VCLK3 domain off. The VCLK3 domain is enabled.
		1	The VCLK3 domain is disabled.
7	Reserved		Reads return zero or one and privilege mode writes allowed.
7-6	RTICK1OFF	0	RTICK1 domain off. The RTICK1 domain is enabled.
		1	The RTICK1 domain is disabled.
5-4	VCLKA[2-1]OFF	0	VCLKA[2-1] domain off. The VCLKA[2-1] domain is enabled.
		1	The VCLKA[2-1] domain is disabled.
3	VCLK2OFF	0	VCLK2 domain off. The VCLK2 domain is enabled.
		1	The VCLK2 domain is disabled.
2	VCLKPOFF	0	VCLK_periph domain off. The VCLK_periph domain is enabled.
		1	The VCLK_periph domain is disabled.
1	HCLKOFF	0	HCLK and VCLK_sys domains off. The HCLK and VCLK_sys domains are enabled.
		1	The HCLK and VCLK_sys domains are disabled.
0	GCLKOFF	0	GCLK domain off. The GCLK domain is enabled.
		1	The GCLK domain is disabled.





**Table 2-33. Clock Domain Disable Set Register (CDDISSET) Field Descriptions (continued)**

Bit	Field	Value	Description
3	SETVCLK2OFF	0 1	Set VCLK2 domain. <i>Read:</i> The VCLK2 domain is enabled. <i>Write:</i> The VCLK2 domain is unchanged. <i>Read:</i> The VCLK2 domain is disabled. <i>Write:</i> The VCLK2 domain is set to the enabled state.
2	SETVCLKPOFF	0 1	Set VCLK_periph domain. <i>Read:</i> The VCLK_periph domain is enabled. <i>Write:</i> The VCLK_periph domain is unchanged. <i>Read:</i> The VCLK_periph domain is disabled. <i>Write:</i> The VCLK_periph domain is set to the enabled state.
1	SETHCLKOFF	0 1	Set HCLK and VCLK_sys domains. <i>Read:</i> The HCLK and VCLK_sys domain is enabled. <i>Write:</i> The HCLK and VCLK_sys domain is unchanged. <i>Read:</i> The HCLK and VCLK_sys domain is disabled. <i>Write:</i> The HCLK and VCLK_sys domain is set to the enabled state.
0	SETGCLKOFF	0 1	Set GCLK domain. <i>Read:</i> The GCLK domain is enabled. <i>Write:</i> The GCLK domain is unchanged. <i>Read:</i> The GCLK domain is disabled. <i>Write:</i> The GCLK domain is set to the enabled state.

### 2.5.1.15 Clock Domain Disable Clear Register (CDDISCLR)

The CDDISCLR register, shown in [Figure 2-20](#) and described in [Table 2-34](#), clears clock domains to the enabled state.

**Figure 2-20. Clock Domain Disable Clear Register (CDDISCLR) [offset = 44h]**

	Reserved								
R-0									
15	Reserved			12	11	10	9	8	
	R-0			R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
	7	6	5	4	3	2	1	0	
	Reserved	CLRRTI1CLKO FF	CLRVLKA2O FF	CLRVLKA1O FF	CLRVLK2OF F	CLRVLKPOF F	CLRHCLKOFF	CLRCLK3OF F	
	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	

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

**Table 2-34. Clock Domain Disable Clear Register (CDDISCLR) Field Descriptions**

Bit	Field	Value	Description
31-12	Reserved	0	Read returns 0. Writes have no effect.
11-10	CLRVLKA[4-3]OFF	0	Clear VCLKA[4-3] domain. <i>Read:</i> The VCLKA[4-3] domain is enabled. <i>Write:</i> The VCLKA[4-3] domain is unchanged.
		1	<i>Read:</i> The VCLKA[4-3] domain is disabled. <i>Write:</i> The VCLKA[4-3] domain is cleared to the enabled state.
9	Reserved	0	Reads return zero or one and privilege mode writes allowed.
8	CLRCLK3OFF	0	Clear VCLK3 domain. <i>Read:</i> The VCLK3 domain is enabled. <i>Write:</i> The VCLK3 domain is unchanged.
		1	<i>Read:</i> The VCLK3 domain is disabled. <i>Write:</i> The VCLK3 domain is cleared to the enabled state.
7	Reserved	0	Reads return zero or one and privilege mode writes allowed.
6	CLRRTI1CLKOFF	0	Clear RTICK1 domain. <i>Read:</i> The RTICK1 domain is enabled. <i>Write:</i> The RTICK1 domain is unchanged.
		1	<i>Read:</i> The RTICK1 domain is disabled. <i>Write:</i> The RTICK1 domain is cleared to the enabled state.
5	CLRTVCLKA2OFF	0	Clear VCLKA2 domain. <i>Read:</i> The VCLKA2 domain is enabled. <i>Write:</i> The VCLKA2 domain is unchanged.
		1	<i>Read:</i> The VCLKA2 domain is disabled. <i>Write:</i> The VCLKA2 domain is cleared to the enabled state.
4	CLRVLKA1OFF	0	Clear VCLKA1 domain. <i>Read:</i> The VCLKA1 domain is enabled. <i>Write:</i> The VCLKA1 domain is unchanged.
		1	<i>Read:</i> The VCLKA1 domain is disabled. <i>Write:</i> The VCLKA1 domain is cleared to the enabled state.

**Table 2-34. Clock Domain Disable Clear Register (CDDISCLR) Field Descriptions (continued)**

Bit	Field	Value	Description
3	CLRVCLK2OFF	0 1	Clear VCLK2 domain. <i>Read:</i> The VCLK2 domain is enabled. <i>Write:</i> The VCLK2 domain is unchanged. <i>Read:</i> The VCLK2 domain is disabled. <i>Write:</i> The VCLK2 domain is cleared to the enabled state.
2	CLRVCLKPOFF	0 1	Clear VCLK_periph domain. <i>Read:</i> The VCLK_periph domain is enabled. <i>Write:</i> The VCLK_periph domain is unchanged. <i>Read:</i> The VCLK_periph domain is disabled. <i>Write:</i> The VCLK_periph domain is cleared to the enabled state.
1	CLRHCLKOFF	0 1	Clear HCLK and VCLK_sys domains. <i>Read:</i> The HCLK and VCLK_sys domain is enabled. <i>Write:</i> The HCLK and VCLK_sys domain is unchanged. <i>Read:</i> The HCLK and VCLK_sys domain is disabled. <i>Write:</i> The HCLK and VCLK_sys domain is cleared to the enabled state.
0	CLRGCLKOFF	0 1	Clear GCLK domain. <i>Read:</i> The GCLK domain is enabled. <i>Write:</i> The GCLK domain is unchanged. <i>Read:</i> The GCLK domain is disabled. <i>Write:</i> The GCLK domain is cleared to the enabled state.

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

The GHVSRC register, shown in [Figure 2-21](#) and described in [Table 2-35](#), controls the clock source configuration for the GCLK, HCLK, VCLK and VCLK2 clock domains.

**Figure 2-21. GCLK, HCLK, VCLK, and VCLK2 Source Register (GHVSRC) [offset = 48h]**

31	28	27	24	23	20	19	16	
Reserved		GHVWAKE		Reserved		HVLPM		
R-0		R/W-0		R-0		R/W-0		
15						4	3	0
Reserved						GHVSRC		
R-0						R/W-0		

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

**Table 2-35. GCLK, HCLK, VCLK, and VCLK2 Source Register (GHVSRC) Field Descriptions**

Bit	Field	Value	Description
31-28	Reserved	0	Read returns 0. Writes have no effect.
27-24	GHVWAKE	0 1h 2h 3h 4h 5h 6h 7h 8h-Fh	GCLK, HCLK, VCLK, VCLK2 source on wakeup. Clock source0 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup. Clock source1 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup. Clock source2 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup. Clock source3 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup. Clock source4 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup. Clock source5 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup. Clock source6 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup. Clock source7 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup. Reserved - These values should not be used.
23-20	Reserved	0	Read returns 0. Writes have no effect.
19-16	HVLPM	0 1h 2h 3h 4h 5h 6h 7h 8h-Fh	HCLK, VCLK, VCLK2 source on wakeup when GCLK is turned off. Clock source0 is the source for HCLK, VCLK, VCLK2 on wakeup. Clock source1 is the source for HCLK, VCLK, VCLK2 on wakeup. Clock source2 is the source for HCLK, VCLK, VCLK2 on wakeup. Clock source3 is the source for HCLK, VCLK, VCLK2 on wakeup. Clock source4 is the source for HCLK, VCLK, VCLK2 on wakeup. Clock source5 is the source for HCLK, VCLK, VCLK2 on wakeup. Clock source6 is the source for HCLK, VCLK, VCLK2 on wakeup. Clock source7 is the source for HCLK, VCLK, VCLK2 on wakeup. Reserved - These values should not be used.
15-4	Reserved	0	Read returns 0. Writes have no effect.
3-0	GHVSRC	0 1h 2h 3h 4h 5h 6h 7h 8h-Fh	GCLK, HCLK, VCLK, VCLK2 current source. <b>Note: The GHVSRC[3-0] bits are updated with the HVLPM[3-0] setting when GCLK is turned off, and are updated with the GHVWAKE[3-0] setting on system wakeup.</b> Clock source0 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup. Clock source1 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup. Clock source2 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup. Clock source3 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup. Clock source4 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup. Clock source5 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup. Clock source6 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup. Clock source7 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup. Reserved - These values should not be used.

---

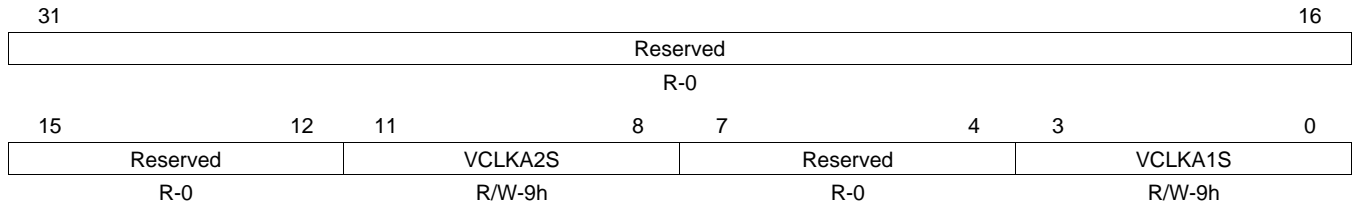
**NOTE:** Non implemented clock sources should not be enabled or used. A list of the available clock sources is shown in the [Clock Sources Table](#).

---

### 2.5.1.17 Peripheral Asynchronous Clock Source Register (VCLKASRC)

The VCLKASRC register, shown in [Figure 2-22](#) and described in [Table 2-36](#), sets the clock source for the asynchronous peripheral clock domains to be configured to run from a specific clock source.

**Figure 2-22. Peripheral Asynchronous Clock Source Register (VCLKASRC) [offset = 4Ch]**



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

**Table 2-36. Peripheral Asynchronous Clock Source Register (VCLKASRC) Field Descriptions**

Bit	Field	Value	Description
31-12	Reserved	0	Read returns 0. Writes have no effect.
11-8	VCLKA2S	0	Peripheral asynchronous clock2 source.
		1h	Clock source0 is the source for peripheral asynchronous clock2.
		2h	Clock source1 is the source for peripheral asynchronous clock2.
		3h	Clock source2 is the source for peripheral asynchronous clock2.
		4h	Clock source3 is the source for peripheral asynchronous clock2.
		5h	Clock source4 is the source for peripheral asynchronous clock2.
		6h	Clock source5 is the source for peripheral asynchronous clock2.
		7h	Clock source6 is the source for peripheral asynchronous clock2.
		8h-Fh	Clock source7 is the source for peripheral asynchronous clock2.
		8h-Fh	VCLK is the source for peripheral asynchronous clock2.
7-4	Reserved	0	Read returns 0. Writes have no effect.
3-0	VCLKA1S	0	Peripheral asynchronous clock1 source.
		1h	Clock source0 is the source for peripheral asynchronous clock1.
		2h	Clock source1 is the source for peripheral asynchronous clock1.
		3h	Clock source2 is the source for peripheral asynchronous clock1.
		4h	Clock source3 is the source for peripheral asynchronous clock1.
		5h	Clock source4 is the source for peripheral asynchronous clock1.
		6h	Clock source5 is the source for peripheral asynchronous clock1.
		7h	Clock source6 is the source for peripheral asynchronous clock1.
		8h-Fh	Clock source7 is the source for peripheral asynchronous clock1.
		8h-Fh	VCLK is the source for peripheral asynchronous clock1.

**NOTE:** Non implemented clock sources should not be enabled or used. A list of the available clock sources is shown in the Clock Sources Table.



### 2.5.1.18 RTI Clock Source Register (RCLKSRC)

The RCLKSRC register, shown in [Figure 2-23](#) and described in [Table 2-37](#), controls the RTI (Real Time Interrupt) clock source selection.

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

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

**Figure 2-23. RTI Clock Source Register (RCLKSRC) [offset = 50h]**

31	26	25	24	23	20	19	16
Reserved				Reserved		Reserved	
R-0				R/W-1h		R/W-9h	
15	10	9	8	7	4	3	0
Reserved				RTI1DIV		RTI1SRC	
R-0				R/W-1h		R/W-9h	

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

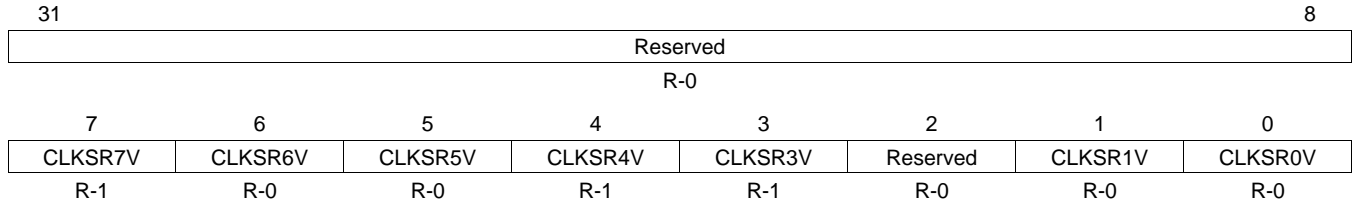
**Table 2-37. RTI Clock Source Register (RCLKSRC) Field Descriptions**

Bit	Field	Value	Description
31-26	Reserved	0	Read returns 0. Writes have no effect.
25-24	Reserved	0	Reads return value and privilege mode writes allowed.
23-20	Reserved	0	Read returns 0. Writes have no effect.
19-16	Reserved	0	Reads return value and privilege mode writes allowed.
15-10	Reserved	0	Read returns 0. Writes have no effect.
9-8	RTI1DIV	0	RTI clock1 Divider. RTICK1 divider value is 1.
		1h	RTICK1 divider value is 2.
		2h	RTICK1 divider value is 4.
		3h	RTICK1 divider value is 8.
7-4	Reserved	0	Read returns 0. Writes have no effect.
3-0	RTI1SRC	0	RTI clock1 source. Clock source0 is the source for RTICK1.
		1h	Clock source1 is the source for RTICK1.
		2h	Clock source2 is the source for RTICK1.
		3h	Clock source3 is the source for RTICK1.
		4h	Clock source4 is the source for RTICK1.
		5h	Clock source5 is the source for RTICK1.
		6h	Clock source6 is the source for RTICK1.
		7h	Clock source7 is the source for RTICK1.
8h-Fh	VCLK is the source for RTICK1.		

**NOTE:** A list of the available clock sources is shown in the [Clock Sources Table](#).

**2.5.1.19 Clock Source Valid Status Register (CSVSTAT)**

The CSVSTAT register, shown in [Figure 2-24](#) and described in [Table 2-38](#), indicates the status of usable clock sources.

**Figure 2-24. Clock Source Valid Status Register (CSVSTAT) [offset = 54h]**


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

**Table 2-38. Clock Source Valid Register (CSVSTAT) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved.	0	Reads return 0. Writes have no effect.
7-3	CLKSR[7-3]V	0	Clock source[7-0] valid.
		1	Clock source[7-0] is not valid.
			Clock source[7-0] is valid.
			<b>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 Global Clock Module (GCM).</b>
2	Reserved.	0	Reads return 0. Writes have no effect.
1-0	CLKSR[1-0]V	0	Clock source[1-0] valid.
		1	Clock source[1-0] is not valid.
			Clock source[1-0] is valid.
			<b>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.</b>

**NOTE:** A list of the available clock sources is shown in the [Clock Sources Table](#).

### 2.5.1.20 Memory Self-Test Global Control Register (MSTGCR)

The MSTGCR register, shown in [Figure 2-25](#) and described in [Table 2-39](#), controls several aspects of the PBIST (Programmable Built-In Self Test) memory controller.

**Figure 2-25. Memory Self-Test Global Control Register (MSTGCR) [offset = 58h]**

31	24	23	16
Reserved		MBIST_ALGSEL	
R-0		R/W-0	
15	10	9	8
Reserved		ROM_DIV	Reserved
R-0		R/W-0	R-0
7	4	3	0
MSTGENA			R/W-5h

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

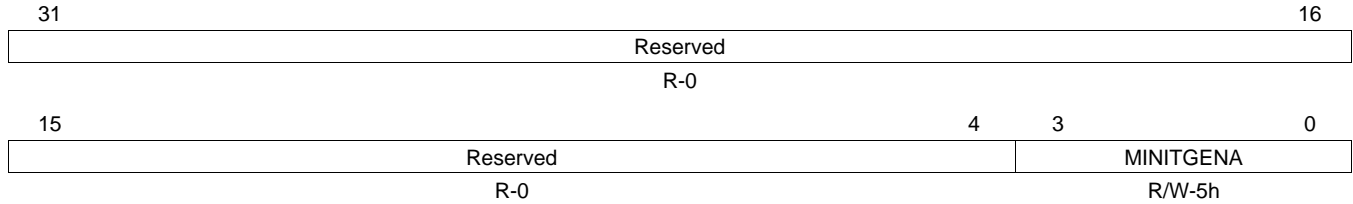
**Table 2-39. Memory Self-Test Global Control Register (MSTGCR) Field Descriptions**

Bit	Field	Value	Description
31-24	Reserved	0	Read returns 0. Writes have no effect.
23-16	MBIST_ALGSEL	0 1h 2h 4h 8h 10h 20h 40h 80h	Selects different test algorithm for PBIST No Algorithm selected. Checkerboard has been selected. March 13N with background of all 0s and all 1s has been selected. March 11N with background of hex 5 and As has been selected. March 13N with backgrounds of hex 3 and Cs, hex 0F and F0s, and 69 and 96s has been selected. PMOS Open Address Decode has been selected. No Algorithm selected. No Algorithm selected. No Algorithm selected.
15-10	Reserved	0	Read returns 0. Writes have no effect.
9-8	ROM_DIV	0 1h 2h 3h	Prescaler divider bits for ROM clock source. ROM clock source is HCLK divided by 1. PBIST will reset for 16 VBUS cycles. ROM clock source is HCLK divided by 2. PBIST will reset for 32 VBUS cycles. ROM clock source is HCLK divided by 4. PBIST will reset for 64 VBUS cycles. ROM clock source is HCLK divided by 8. PBIST will reset for 96 VBUS cycles.
7-4	Reserved	0	Read returns 0. Writes have no effect.
3-0	MSTGENA	Ah Others	Memory self-test controller global enable key <b>Note: Enabling the MSTGENA key will generate a reset to the state machine of the selected PBIST controller.</b> Memory self-test controller is enabled. Memory self-test controller is disabled. <b>Note: It is recommended that a value of 0101b be used to disable the memory self-test controller. This value will give maximum protection from a bit flip ending event that would inadvertently enable the controller.</b>

**2.5.1.21 Memory Hardware Initialization Global Control Register (MINITGCR)**

The MINITGCR register, shown in [Figure 2-26](#) and described in [Table 2-40](#), enables automatic hardware memory initialization.

**Figure 2-26. Memory Hardware Initialization Global Control Register (MINITGCR) [offset = 5Ch]**



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

**Table 2-40. Memory Hardware Initialization Global Control Register (MINITGCR) Field Descriptions**

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

### 2.5.1.22 MBIST Controller/ Memory Initialization Enable Register (MSINENA)

The MSINENA register, shown in [Figure 2-27](#) and described in [Table 2-41](#), enables PBIST controllers for memory self test and the memory modules initialized during automatic hardware memory initialization.

**Figure 2-27. MBIST Controller/Memory Initialization Enable Register (MSINENA) [offset = 60h]**

31	MSIENA	16
	R/W-0	
15	MSIENA	0
	R/W-0	

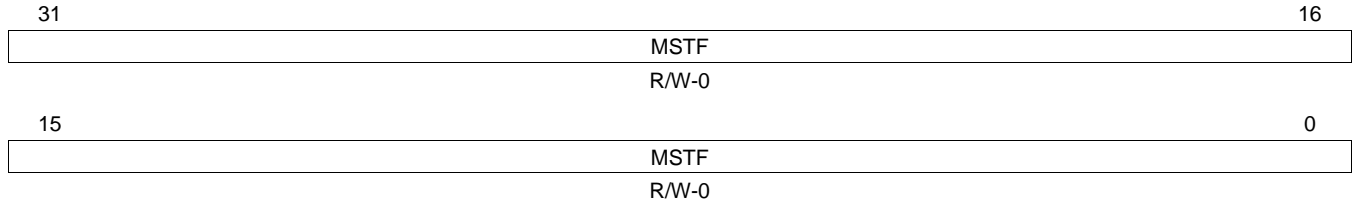
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 2-41. MBIST Controller/Memory Initialization Enable Register (MSINENA) Field Descriptions**

Bit	Field	Value	Description
31-0	MSIENA		<p>PBIST controller and memory initialization enable register.</p> <p>In memory self-test mode, all the corresponding bits of the memories to be tested should be set before enabling the global memory self-test controller key (MSTGENA) in the MSTGCR register (offset 58h). The reason for this is that MSTGENA, in addition to being the global enable for all individual PBIST controllers, is the source for the reset generation to all the PBIST controller state machines.</p> <p>Disabling the MSTGENA or MINITGENA key (by writing from a Ah to any other value) will reset all the MSIENA[31-0] bits to their default values.</p> <p>0  <i>In memory self-test mode (MSTGENA = Ah):</i>            PBIST controller [31-0] is disabled.</p> <p><i>In memory Initialization mode (MINITGENA = Ah):</i>            Memory module [31-0] auto hardware initialization is disabled.</p> <p>1  <i>In memory self-test mode (MSTGENA = Ah):</i>            PBIST controller [31-0] is enabled.</p> <p><i>In memory Initialization mode (INITGENA = Ah):</i>            Memory module[31-0] auto hardware initialization is enabled.</p> <p><b>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.</b></p>

**2.5.1.23 Memory Self-Test Fail Status Register (MSTFAIL)**

The MSTFAIL register, shown in [Figure 2-28](#) and described in [Table 2-42](#), shows the fail status of the memory self-tests.

**Figure 2-28. Memory Self-Test Fail Status Register (MSTFAIL) [offset = 64h]**


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

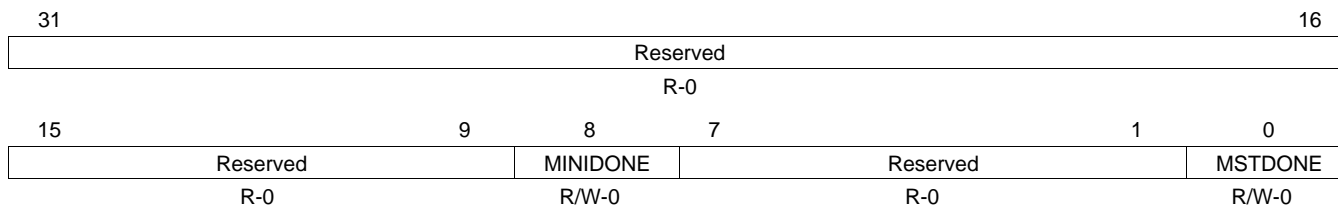
**Table 2-42. Memory Self-Test Fail Status Register (MSTFAIL) Field Descriptions**

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

### 2.5.1.24 MSTC Global Status Register (MSTCGSTAT)

The MSTCGSTAT register, shown in Figure 2-29 and described in Table 2-43, shows the status of the memory hardware initialization and the memory self-test.

**Figure 2-29. MSTC Global Status Register (MSTCGSTAT) [offset = 68h]**



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

**Table 2-43. MSTC Global Status Register (MSTCGSTAT) Field Descriptions**

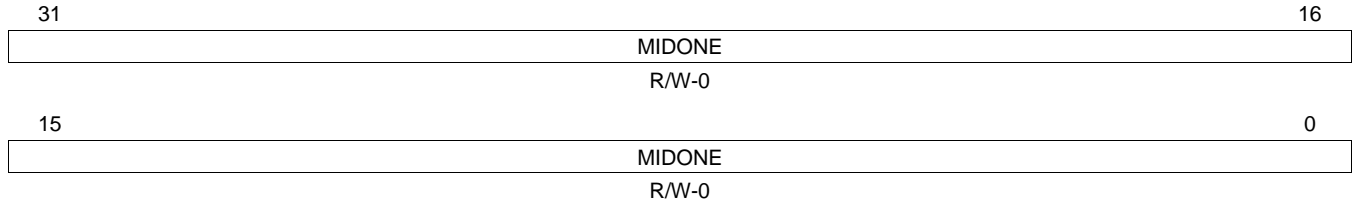
Bit	Field	Value	Description
31-9	Reserved	0	Read returns 0. Writes have no effect.
8	MINIDONE	0	Memory hardware initialization complete status. <b>Note: Disabling the MINITGENA key (By writing from a Ah to any other value) will clear the MINIDONE status bit to 0.</b> <b>Note: Individual memory initialization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



### 2.5.1.25 Memory Hardware Initialization Status Register (MINISTAT)

The MINISTAT register, shown in [Figure 2-30](#) and described in [Table 2-44](#), indicates the status of hardware memory initialization.

**Figure 2-30. Memory Hardware Initialization Status Register (MINISTAT) [offset = 6Ch]**



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

**Table 2-44. Memory Hardware Initialization Status Register (MINISTAT) Field Descriptions**

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

### 2.5.1.26 PLL Control Register 1 (PLLCTL1)

The PLLCTL1 register, shown in Figure 2-31 and described in Table 2-45, controls the output frequency of PLL1 (Clock Source 1 - FMzPLL). It also controls the behavior of the device if a PLL slip or oscillator failure is detected.

**Figure 2-31. PLL Control Register 1 (PLLCTL1) [offset = 70h]**

31	30	29	28	24	23	22	21	16
ROS	MASK_SLIP	PLLDIV			ROF	Reserved	REFCLKDIV	
R/W-0	R/W-1h	R/W-Fh			R/W-0	R-0	R/W-2h	
15								0
PLLMUL								
R/W-5F00h								

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

**Table 2-45. PLL Control Register 1 (PLLCTL1) Field Descriptions**

Bit	Field	Value	Description
31	ROS	0 1	Reset on PLL Slip Do not reset system when PLL slip is detected Reset when PLL slip is detected <b>Note: BPOS (Bits 30-29) must also be enabled for ROS to be enabled.</b>
30-29	MASK_SLIP	2h Others	Mask detection of PLL slip If a PLL Slip is detected no action is taken. 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. <b>Note: If ROS (Bit 31) is set to 1 the device will be reset if a PLL Slip and the PLL will be bypassed after the reset occurs.</b>
28-24	PLLDIV	0 to 31 0 1h : 1Fh	PLL Output Clock Divider $R = PLLDIV + 1$ $f_{PLL\ CLK} = f_{post-ODCLK} / R$ $f_{PLL\ CLK} = f_{post-ODCLK} / 1$ $f_{PLL\ CLK} = f_{post-ODCLK} / 2$ continues in sequence $f_{PLL\ CLK} = f_{post-ODCLK} / 32$
23	ROF	0 1	Reset on Oscillator Fail Do not reset system when oscillator is out of range. Reset when oscillator is out of range.
22	Reserved	0	Read/Write, but value has no effect on PLL operation.
21-16	REFCLKDIV	0 to 63 0 1h : 3Fh	Reference Clock Divider $NR = REFCLKDIV + 1$ $f_{INT\ CLK} = f_{OSCIN} / NR$ $f_{INT\ CLK} = f_{OSCIN} / 1$ $f_{INT\ CLK} = f_{OSCIN} / 2$ continues in sequence $f_{INT\ CLK} = f_{OSCIN} / 64$
15-0	PLLMUL	5B00h 5C00h : B700h	PLL Multiplication Factor Valid multiplication factors are from 92 to 184. $NF = (PLLMUL / 256) + 1$ $f_{VCO\ CLK} = f_{INT\ CLK} \times NF$ $f_{VCO\ CLK} = f_{INT\ CLK} \times 92$ $f_{VCO\ CLK} = f_{INT\ CLK} \times 93$ continues in sequence $f_{VCO\ CLK} = f_{INT\ CLK} \times 184$

### 2.5.1.27 PLL Control Register 2 (PLLCTL2)

The PLLCTL2 register, shown in [Figure 2-32](#) and described in [Table 2-46](#), controls the modulation characteristics and the output divider of the PLL.

**NOTE: Modulation Frequency and Depth Setting Constraints**

There are several combinations of the modulation depth and modulation frequency that are not allowed. Some of these settings effect the PLL even when frequency modulation is not enabled. Refer the device datasheet to identify these combinations to avoid PLL malfunction.

**Figure 2-32. PLL Control Register 2 (PLLCTL2) [offset = 74h]**

31	30	22	21	20	16
FMENA	SPREADINGRATE			Reserved	MULMOD
R/W-0	R/W-1FFh			R-0	R/W-7h
15	12	11	9	8	0
MULMOD		ODPLL		SPR_AMOUNT	
R/W-7h		R/W-1h		R/W-0	

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

**Table 2-46. PLL Control Register 2 (PLLCTL2) Field Descriptions**

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

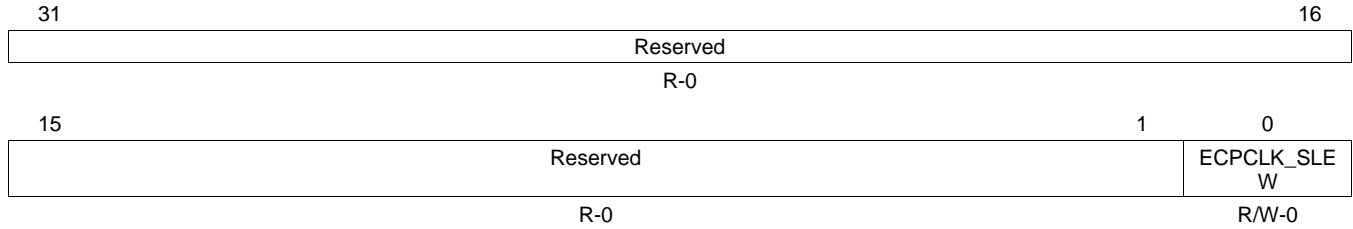
**Table 2-46. PLL Control Register 2 (PLLCTL2) Field Descriptions (continued)**

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

**2.5.1.28 SYS Pin Control Register 10 (SYSPC10)**

The SYSPC10 register, shown in [Figure 2-33](#) and described in [Table 2-47](#), controls the function of the ECPClk slew mode.

**Figure 2-33. SYS Pin Control Register 10 (SYSPC10) [offset = 78h]**



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

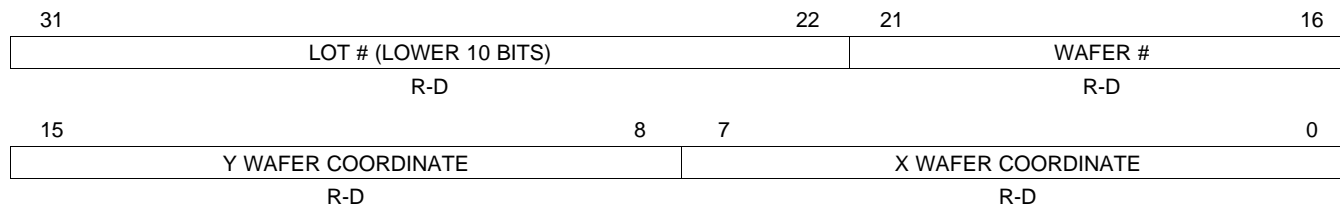
**Table 2-47. SYS Pin Control Register 10 (SYSPC10) Field Descriptions**

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

### 2.5.1.29 Die Identification Register Lower Word (DIEIDL)

The DIEIDL register, shown in [Figure 2-34](#) and described in [Table 2-48](#), contains information about the die lot number, wafer number, and X, Y wafer coordinates.

**Figure 2-34. Die Identification Register, Lower Word (DIEIDL) [offset = 7Ch]**



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

**Table 2-48. Die Identification Register, Lower Word (DIEIDL) Field Descriptions**

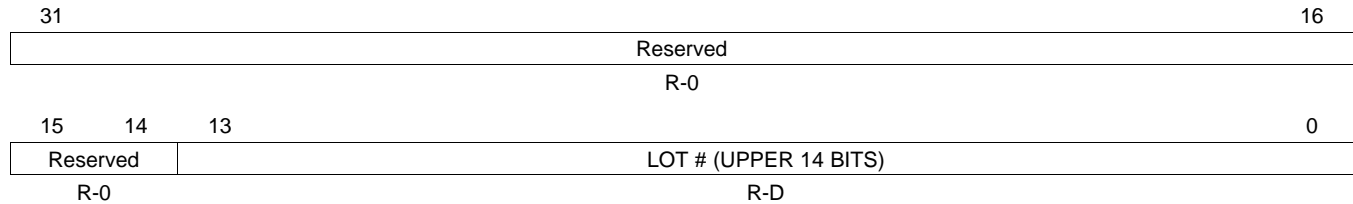
Bit	Field	Value	Description
31-22	LOT # (Lower 10 Bits)	-	These read only bits contain the lower 10 bits of the device lot number.
21-16	WAFER #	-	These read only bits contain the wafer number of the device.
15-8	Y WAFER COORDINATE	-	These read only bits contain the Y wafer coordinate of the device
7-0	X WAFER COORDINATE	-	These read only bits contain the X wafer coordinate of the device

**NOTE: Die Identification Information**

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

**2.5.1.30 Die Identification Register Upper Word (DIEIDH)**

The DIEIDH register, shown in [Figure 2-35](#) and described in [Table 2-49](#), contains information about the die lot number.

**Figure 2-35. Die Identification Register, Upper Word (DIEIDH) [offset = 80h]**


LEGEND: R/W = Read/Write; R = Read only; D = Value is device dependent; -n = value after reset

**Table 2-49. Die Identification Register, Upper Word (DIEIDH) Field Descriptions**

Bit	Field	Value	Description
31-14	Reserved	0	Read returns 0. Writes have no effect.
13-0	LOT # (Upper 14 Bits)		This read-only register contains the upper 14 bits of the device lot number.

---

**NOTE: Die Identification Information**

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

---



### 2.5.1.31 LPO/Clock Monitor Control Register (LPOMONCTL)

The LPOMONCTL register, shown in [Figure 2-36](#) and described in [Table 2-50](#), controls the Low Frequency (Clock Source 4) and High Frequency (Clock Source 5) Low Power Oscillator's trim values.

**Figure 2-36. LPO/Clock Monitor Control Register (LPOMONCTL) [offset = 088h]**

31	25	24	23	17	16	
Reserved		BIAS ENABLE	Reserved		OSCFRQ CONFIGCNT	
R-0		R/W-1	R-0		R/W-0	
15	13	12	8	7	5 4 0	
Reserved		HFTRIM		Reserved		LFTRIM
R-0		R/W-10h		R-0		R/W-10h

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

**Table 2-50. LPO/Clock Monitor Control Register (LPOMONCTL) Field Descriptions**

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

**Table 2-50. LPO/Clock Monitor Control Register (LPOMONCTL) Field Descriptions (continued)**

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

**Table 2-50. LPO/Clock Monitor Control Register (LPOMONCTL) Field Descriptions (continued)**

Bit	Field	Value	Description
4-0	LFTRIM		<p>Low frequency oscillator trim value. This four-bit value is used to center the LF oscillator's frequency.</p> <p><b>Caution: This value should only be changed when the LF oscillator is not the source for a clock domain, otherwise a system failure could result.</b></p> <p>The values below are the ratio: <math>f / f_0</math> in the F021 process.</p>
		0	20.67
		1h	25.76
		2h	30.84
		3h	35.90
		4h	40.93
		5h	45.95
		6h	50.97
		7h	55.91
		8h	60.86
		9h	65.78
		Ah	70.75
		Bh	75.63
		Ch	80.61
		Dh	85.39
		Eh	90.23
		Fh	95.11
		10h	100.00% Default at Reset
		11h	104.84
		12h	109.51
		13h	114.31
		14h	119.01
		15h	123.75
		16h	128.62
		17h	133.31
		18h	138.03
		19h	142.75
		1Ah	147.32
		1Bh	152.02
		1Ch	156.63
		1Dh	161.38
		1Eh	165.90
		1Fh	170.42

### 2.5.1.32 Clock Test Register (CLKTEST)

The CLKTEST register, shown in [Figure 2-37](#) and described in [Table 2-51](#), controls the clock signal that is supplied to the ECLK pin for test and debug purposes.

---

**NOTE: Clock Test Register Usage**

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

---

**Figure 2-37. Clock Test Register (CLKTEST) [offset = 8Ch]**

31	27	26	25	24
Reserved		ALTLIMPCLOCK ENABLE	RANGEDET CTRL	RANGEDETEN ASSEL
R-0		R/W-0	R/W-0	R/W-0
23	20	19	16	
Reserved		CLK_TEST_EN		
R-0		R/W-Ah		
15	12	11	8	7
Reserved		SEL_GIO_PIN	Reserved	
R-0		R/W-0	R-0	
		4	3	0
		SEL_ECP_PIN		
		R/W-0		

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

**Table 2-51. Clock Test Register (CLKTEST) Field Descriptions**

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

**Table 2-51. Clock Test Register (CLKTEST) Field Descriptions (continued)**

Bit	Field	Value	Description
11-8	SEL_GIO_PIN		GIOB[0] pin clock source valid, clock source select
		0	Oscillator valid status
		1h	PLL clock status
		2h	Reserved
		3h	Reserved
		4h	Reserved
		5h	High frequency clock LPO (Low Power Oscillator) clock valid status
		6h	Secondary PLL free-running clock output valid status
		7h	Reserved
		8h	Low frequency clock LPO (Low Power Oscillator) clock
		9h	Oscillator valid status
		Ah	Oscillator valid status
		Bh	Oscillator valid status
		Ch	Oscillator valid status
		Dh	Oscillator valid status
Eh	Oscillator valid status		
Fh	Oscillator valid status		
7-4	Reserved	0	Read returns 0. Writes have no effect.
3-0	SEL_ECP_PIN		ECLK pin clock source select <b>Note: Only valid clock sources can be selected for the ECLK pin. Valid clock sources are displayed by the CSVSTAT register.</b>
		0	Oscillator clock
		1h	PLL clock
		2h	Not Implemented
		3h	EXTCLKIN1
		4h	Low frequency LPO (Low Power Oscillator) clock
		5h	High frequency clock LPO (Low Power Oscillator) clock
		6h	Secondary PLL free-running clock output
		7h	EXCLKIN2
		8h	GCLK
		9h	RTI Base
		Ah	Not Implemented
		Bh	VCLKA1
		Ch	VCLKA2
		Dh	VCLKA3_S
Eh	VCLKA4		
Fh	Flash HD Pump Oscillator		

---

**NOTE:** Non implemented clock sources should not be enabled or used.

---

### 2.5.1.33 DFT Control Register (DFTCTRLREG)

This register is shown in [Figure 2-38](#) and described in [Table 2-52](#).

**Figure 2-38. DFT Control Register (DFTCTRLREG) [offset = 90h]**

31											16												
Reserved																							
R-0																							
15		14		13		12		11		10		9		8		7		4		3		0	
Reserved		DFTWRITE		Reserved		DFTREAD		Reserved		Reserved		TEST_MODE_KEY											
R-0		R/W-1h		R-0		R/W-1h		R-0		R-0		R-0		R-0		R-0		R-0		R/W-5h		R/W-5h	

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

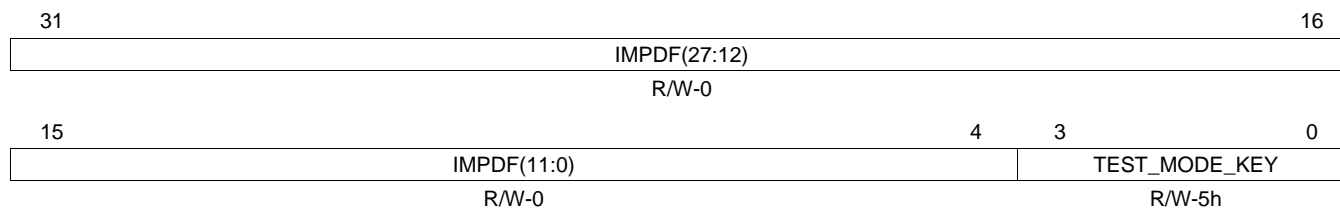
**Table 2-52. DFT Control Register (DFTCTRLREG) Field Descriptions**

Bit	Field	Value	Description
31-14	Reserved	0	Read returns 0. Writes have no effect.
13-12	DFTWRITE		DFT logic access. For F021: DFTWRITE[0] = 0 and DFTREAD[0] = 0 configured in stress mode DFTWRITE[1] = 0 and DFTREAD[1] = 0 configured in stress mode DFTWRITE[0] = 0 and DFTREAD[0] = 0 configured in fast mode DFTWRITE[1] = 1 and DFTREAD[1] = 1 configured in fast mode DFTWRITE[0] = 1 and DFTREAD[0] = 1 configured in slow mode DFTWRITE[1] = 0 and DFTREAD[1] = 0 configured in slow mode DFTWRITE[0] = 1 and DFTREAD[0] = 1 configured in screen mode DFTWRITE[1] = 1 and DFTREAD[1] = 1 configured in screen mode
11-10	Reserved	0	Read returns 0. Writes have no effect.
9-8	DFTREAD		DFT logic access. For F021: DFTWRITE[0] = 0 and DFTREAD[0] = 0 configured in stress mode DFTWRITE[1] = 0 and DFTREAD[1] = 0 configured in stress mode DFTWRITE[0] = 0 and DFTREAD[0] = 0 configured in fast mode DFTWRITE[1] = 1 and DFTREAD[1] = 1 configured in fast mode DFTWRITE[0] = 1 and DFTREAD[0] = 1 configured in slow mode DFTWRITE[1] = 0 and DFTREAD[1] = 0 configured in slow mode DFTWRITE[0] = 1 and DFTREAD[0] = 1 configured in screen mode DFTWRITE[1] = 1 and DFTREAD[1] = 1 configured in screen mode
7-4	Reserved	0	Read returns 0. Writes have no effect.
3-0	TEST_MODE_KEY	0 - Fh (except Ah)  Ah	Test mode key. This register is for internal TI use only. Register key disable. All bits in this register will maintain their default value and cannot be written. Register key enable. ALL the bits can be written to only when the key is enabled. On reset, these bits will be set to 0x5.

### 2.5.1.34 DFT Control Register (DFTCTRLREG2)

This register is shown in [Figure 2-39](#) and described in [Table 2-53](#).

**Figure 2-39. DFT Control Register 2 (DFTCTRLREG2) [offset = 94h]**



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

**Table 2-53. DFT Control Register 2 (DFTCTRLREG2) Field Descriptions**

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



**2.5.1.35 General Purpose Register (GPREG1)**

This register is shown in [Figure 2-40](#) and described in [Table 2-54](#).

**Figure 2-40. General Purpose Register (GPREG1) [offset = A0h]**

31	26	25	20	19	16
Reserved		PLL1_FBSLIP_FILTER_COUNT		PLL1_RFSLIP_FILTER_KEY	
R-0		R/W-0		R/W-5h	
15					0
OUTPUT_BUFFER_LOW_EMI_MODE					
R/W-FFFFh					

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

**Table 2-54. General Purpose Register (GPREG1) Field Descriptions**

Bit	Field	Value	Description
31-26	Reserved	0	Read returns 0. Writes have no effect.
25-20	PLL1_FBSLIP_FILTER_COUNT	0 - 3Fh	FBSLIP down counter programmed value. Configures the system response when a FBSLIP is indicated by the PLL macro. When PLL1_RFSLIP_FILTER_KEY is not 1010, the down counter counts from the programmed value on every LPO high frequency clock once PLL macro indicates FBSLIP. When the count reaches zero, if the synchronized FBSLIP signal is still high, an FBSLIP condition is indicated to the system module and is captured in the global status register. When the FBSLIP signal from the PLL macro is de-asserted before the count reaches 0, the counter is reloaded with the programmed value.
19-16	PLL1_RFSLIP_FILTER_KEY	5h Ah Fh	Configures the system response when a FBSLIP is indicated by the PLL macro. The FBSLIP filter is bypassed and the FBSLIP indicated by the PLL macro is captured in the system module global status register. The FBSLIP filter is enabled and the one-stage synchronization circuit is used. The FBSLIP filter is enabled and the two-stage synchronization circuit is used.
15-0	OUTPUT_BUFFER_LOW_EMI_MODE	0 1h - FFFEh FFFFh	Control field for the low-EMI mode of output buffers for module/signals: bit[0] controls MiBSP11 bit[1] controls SPI2 bit[2] controls MiBSP13 bit[3] controls SPI4 bit[4] controls MiBSP15 Reserved bit[6] controls EMIF bit[7] controls ETM bit[8] controls signal TMS bit[9] controls signal TD1 bit[10] controls signal TD0 bit[11] controls signal RTCK bit[12] controls signal TEST bit[13] controls signal nERROR bit[14] controls signal ADEVT bit[15] controls signal RTP 0 Enable EMI mode for each connected output buffers. 1h - FFFEh Enable/Disable EMI mode for connected output buffers. FFFFh Disable EMI mode for each connected output buffer.

### 2.5.1.36 Imprecise Fault Status Register (IMPFASTS)

The IMPFASTS register, shown in [Figure 2-41](#) and described in [Table 2-55](#), displays information about imprecise aborts that have occurred.

**Figure 2-41. Imprecise Fault Status Register (IMPFASTS) [offset = A8h]**

31											24	23					16
Reserved											MASTERID						
R-0											R-0						
15						11	10	9	8	7				1	0		
Reserved					EMIFA	NCBA	VBUSA	Reserved			ATYPE						
R-0					R-0	R-0	R-0	R-0			R/WC-0						

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

**Table 2-55. Imprecise Fault Status Register (IMPFASTS) Field Descriptions**

Bit	Field	Value	Description
31-24	Reserved	0	Read returns 0. Writes have no effect.
23-16	MASTERID	0-FFh	Master ID. This register indicates which master is responsible for the imprecise abort. The master ID value depends on device implementation- see <a href="#">Table 2-2</a> for MASTERID values for each bus master. <b>Notes:</b> <ul style="list-style-type: none"> <li>• These bits are only updated when an imprecise abort occurs</li> <li>• These bits are cleared to 0 only on power-up reset. The value of these bits remains unchanged after all other resets.</li> </ul>
15-11	Reserved	0	Read returns 0. Writes have no effect.
10	EMIFA	0 1	EMIF imprecise abort. This register indicates the imprecise abort was generated writing into the EMIF. <b>Notes:</b> <ul style="list-style-type: none"> <li>• This bit is only updated when an imprecise abort occurs</li> <li>• This bit is cleared to 0 only on power-up reset. The value of this register remains unchanged after all other resets.</li> </ul> 0 EMIF did not generate the last imprecise abort. 1 EMIF was written with an illegal address and generated an imprecise abort.
9	NCBA	0 1	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. <b>Notes:</b> <ul style="list-style-type: none"> <li>• This bit is only updated when an imprecise abort generated by a non-cacheable, bufferable write or shared device write occurs.</li> <li>• This bit is cleared to 0 only on power-up reset. The value of this register remains unchanged after all other resets.</li> </ul> 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	0 1	VBUS abort. This register indicates the imprecise abort was generated when writing into the peripheral frame. <b>Notes:</b> <ul style="list-style-type: none"> <li>• This bit is only updated when an imprecise abort is generated when writing into the peripheral frame</li> <li>• This bit is cleared to 0 only on power-up reset. The value of this register remains unchanged after all other resets.</li> </ul> 0 The peripheral frame did not generate the last imprecise abort. 1 The peripheral frame was written with an illegal address and generated an imprecise abort.
7-1	Reserved	0	Read returns 0. Writes have no effect.

**Table 2-55. Imprecise Fault Status Register (IMPFASTS) Field Descriptions (continued)**

Bit	Field	Value	Description
0	ATYPE		<p>Abort type. This bit indicates to the CPU whether the last abort was an imprecise abort or a precise abort.</p> <p><b>Notes:</b></p> <ul style="list-style-type: none"> <li>• This bit is updated after each abort is generated to the CPU.</li> <li>• This bit is cleared on CPU read.</li> <li>• This bit is cleared to 0 only on power-up reset. The value of this bit remains unchanged after all other resets</li> </ul> <p>0 The last abort generated was a precise abort. MASTERID, VBUSA, NCBA, EMIFA and IMPFTADD were not updated.</p> <p>1 The last abort generated was an imprecise abort. MASTERID, VBUSA, NCBA, EMIFA and IMPFTADD were updated.</p> <p><b>Note: Once ATYPE is set, the IMPFAWADD and IMPFASTS bits are not updated by subsequent ABORT signals.</b></p>

---

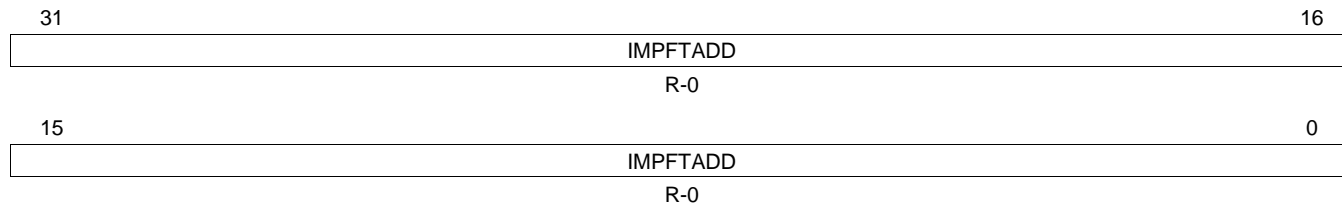
**NOTE:** The DMA, DMM, and the peripheral master port will also generate an imprecise abort to the CPU when writing to the peripheral region or to the EMIF region. This will be indicated in the Master ID field of this register.

---

### 2.5.1.37 Imprecise Fault Address Register (IMPFTADD)

This IMPFTADD register, shown in [Figure 2-42](#) and described in [Table 2-56](#), shows the address at which an imprecise abort occurred.

**Figure 2-42. Imprecise Fault Write Address Register (IMPFTADD) [offset = ACh]**



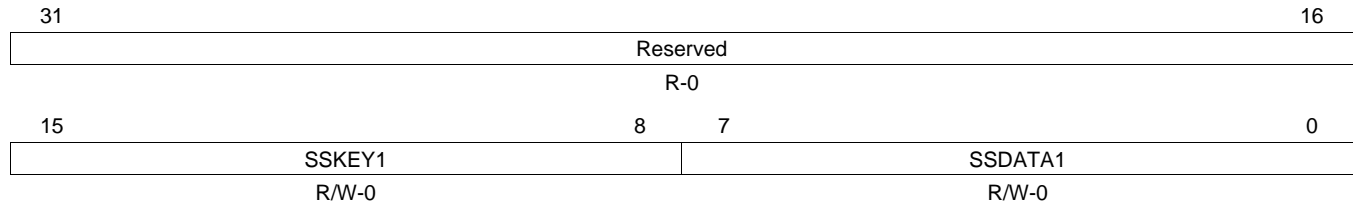
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 2-56. Imprecise Fault Write Address Register (IMPFTADD) Field Descriptions**

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

**2.5.1.38 System Software Interrupt Request 1 Register (SSIR1)**

The SSIR1 register, shown in [Figure 2-43](#) and described in [Table 2-57](#), is used for software interrupt generation.

**Figure 2-43. System Software Interrupt Request 1 Register (SSIR1) [offset = B0h]**


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

**Table 2-57. System Software Interrupt Request 1 Register (SSIR1) Field Descriptions**

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

---

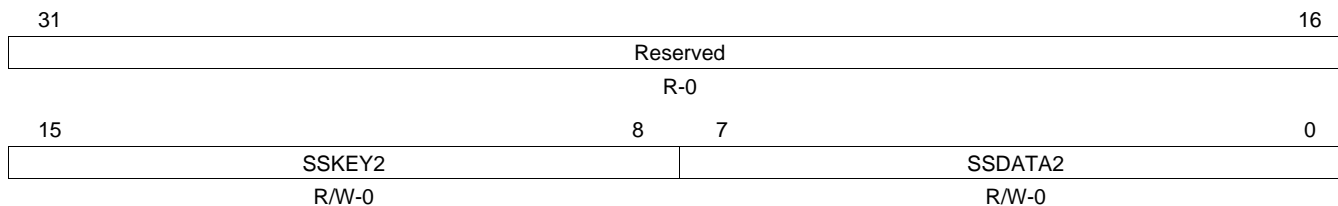
**NOTE:** This register is mirrored at offset FCh for compatibility reasons.

---

### 2.5.1.39 System Software Interrupt Request 2 Register (SSIR2)

The SSIR2 register, shown in [Figure 2-44](#) and described in [Table 2-58](#), is used for software interrupt generation.

**Figure 2-44. System Software Interrupt Request 2 Register (SSIR2) [offset = B4h]**



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

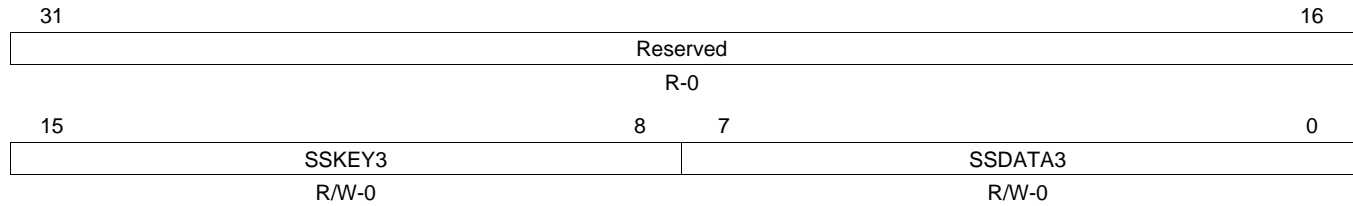
**Table 2-58. System Software Interrupt Request 2 Register (SSIR2) Field Descriptions**

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

### 2.5.1.40 System Software Interrupt Request 3 Register (SSIR3)

The SSIR3 register, shown in [Figure 2-45](#) and described in [Table 2-59](#), is used for software interrupt generation.

**Figure 2-45. System Software Interrupt Request 3 Register (SSIR3) [offset = B8h]**



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

**Table 2-59. System Software Interrupt Request 3 Register (SSIR3) Field Descriptions**

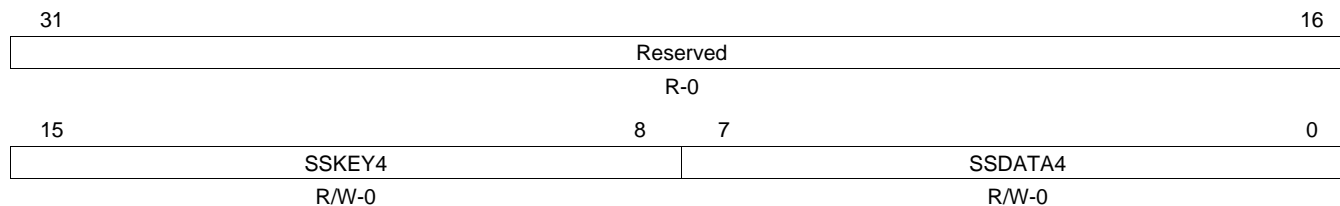
Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-8	SSKEY3	0-FFh	System software interrupt request key. A 93h written to these bits initiates IRQ/FIQ interrupts. Data in this field is always read as 0. The SSKEY3 field can be written into only if the write data matches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



### 2.5.1.41 System Software Interrupt Request 4 Register (SSIR4)

The SSIR4 register, shown in [Figure 2-46](#) and described in [Table 2-60](#), is used for software interrupt generation.

**Figure 2-46. System Software Interrupt Request 4 Register (SSIR4) [offset = BCh]**



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

**Table 2-60. System Software Interrupt Request 4 Register (SSIR4) Field Descriptions**

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

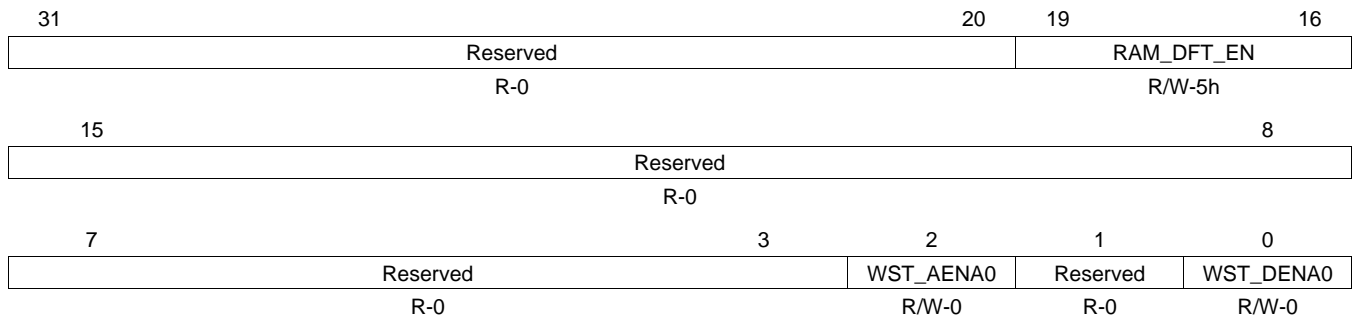
2.5.1.42 RAM Control Register (RAMGCR)

The RAMGCR register, shown in Figure 2-47 and described in Table 2-61, is used to configure eSRAM data and address wait states.

**NOTE: The RAM\_DFT\_EN bits are for TI internal use only.**

The contents of the RAM\_DFT\_EN field should not be changed.

Figure 2-47. RAM Control Register (RAMGCR) [offset = C0h]



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

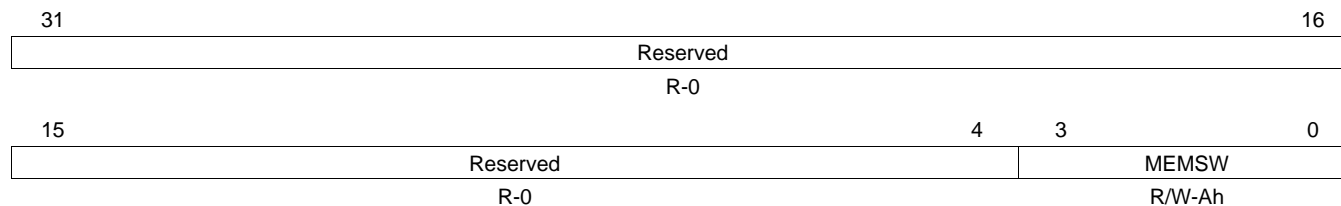
Table 2-61. RAM Control Register (RAMGCR) Field Descriptions

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

### 2.5.1.43 Bus Matrix Module Control Register 1 (BMMCR1)

The BMMCR1 register, shown in [Figure 2-48](#) and described in [Table 2-62](#), allows RAM and Program (Flash) memory addresses to be swapped.

**Figure 2-48. Bus Matrix Module Control Register 1 (BMMCR) [offset = C4h]**



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

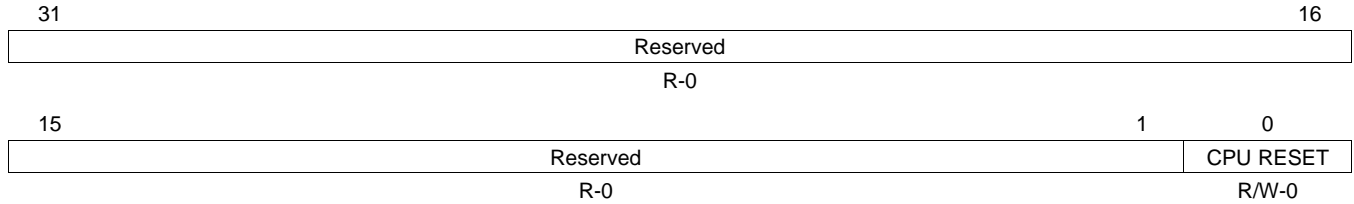
**Table 2-62. Bus Matrix Module Control Register 1 (BMMCR) Field Descriptions**

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

**2.5.1.44 CPU Reset Control Register (CPURSTCR)**

The CPURSTCR register shown in [Figure 2-49](#) and described in [Table 2-63](#), allows the memory protection mode to be configured. It also allows a reset to the Cortex R4 CPU to be generated.

**Figure 2-49. CPU Reset Control Register (CPURSTCR) [offset = CCh]**



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

**Table 2-63. CPU Reset Control Register (CPURSTGCR) Field Descriptions**

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

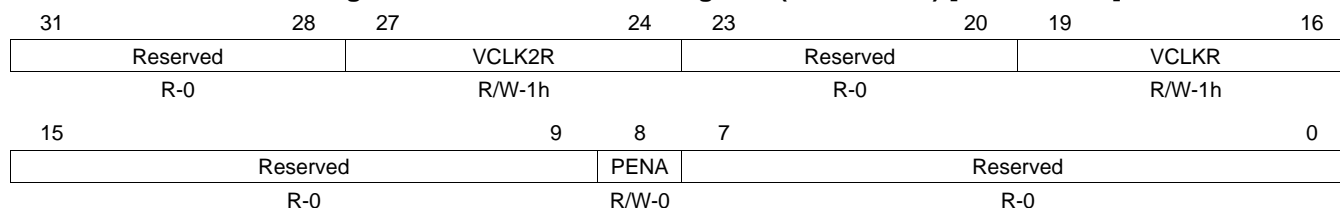
### 2.5.1.45 Clock Control Register (CLKCNTL)

The CLKCNTL register, shown in [Figure 2-50](#) and described in [Table 2-64](#), controls peripheral reset and the peripheral clock divide ratios.

**NOTE: VCLK and VCLK2 clock ratio restrictions.**

VCLK2 must always be greater than or equal to VCLK. In addition, the VCLK and VCLK2 clock ratios must not be changed simultaneously. The application must configure the VCLK2 ratio, read back the contents of the CLKCNTL register, and then configure the VCLK ratio.

**Figure 2-50. Clock Control Register (CLKCNTL) [offset = D0h]**



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

**Table 2-64. Clock Control Register (CLKCNTL) Field Descriptions**

Bit	Field	Value	Description
31-28	Reserved	0	Read returns 0. Writes have no effect.
27-24	VCLK2R	0 ... Fh	VBUS clock2 ratio. The VCLK2 speed is HCLK divided by 1. ... The VCLK2 speed is HCLK divided by 16.
23-20	Reserved	0	Read returns 0. Writes have no effect.
19-16	VCLKR	0 ... Fh	VBUS clock ratio. The VCLK speed is HCLK divided by 1. ... The VCLK speed is HCLK divided by 16.
15-9	Reserved	0	Read returns 0. Writes have no effect.
8	PENA	0 1	Peripheral enable bit. The application must set this bit before accessing any peripheral The global peripheral/peripheral memory frames are in reset. All peripheral/peripheral memory frames are out of reset.
7-0	Reserved	0	Read returns 0. Writes have no effect.

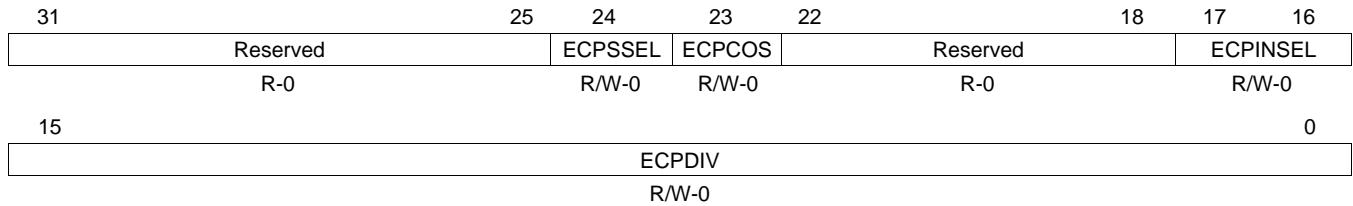
**2.5.1.46 ECP Control Register (ECPCNTL)**

The ECP register, shown in Figure 2-51 and described in Table 2-65, configures the ECLK pin in functional mode.

**NOTE: ECLK Functional mode configuration.**

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

**Figure 2-51. ECP Control Register (ECPCNTL) [offset = D4h]**



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

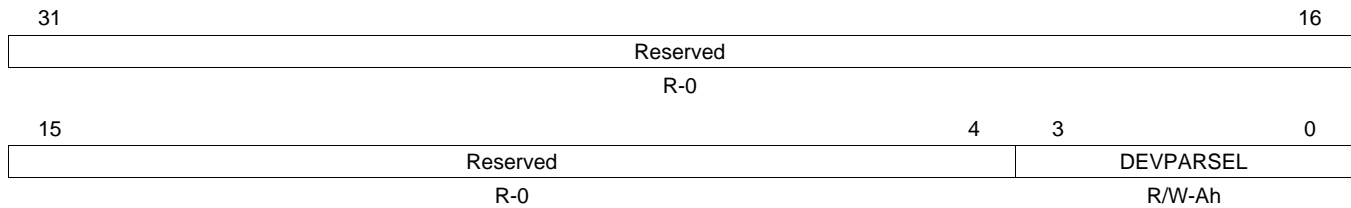
**Table 2-65. ECP Control Register (ECPCNTL) Field Descriptions**

Bit	Field	Value	Description
31-25	Reserved	0	Read returns 0. Writes have no effect.
24	ECPSSEL	0 1	This bit allows the selection between VCLK and OSCIN as the clock source for ECLK. <b>Note: Other ECLK clock sources are available for debug purposes by configuring the CLKTEST register.</b> 0 VCLK is selected as the ECP clock source. 1 OSCIN is selected as the ECP clock source.
23	ECPCOS	0 1	ECP continue on suspend. <b>Note: Suspend mode is entered while performing certain JTAG debugging operations.</b> 0 ECLK output is disabled in suspend mode. ECLK output will be shut off and will not be seen on the I/O pin of the device. 1 ECLK output is not disabled in suspend mode. ECLK output will not be shut off and will be seen on the I/O pin of the device.
22-18	Reserved	0	Read returns 0. Writes have no effect.
17-6	ECPINSEL	0 1h 2h 3h	Select ECP input clock source. 0 Tied Low 1h HCLK 2h External clock 3h Tied Low
15-0	ECPDIV	0-FFFFh	ECP divider value. The value of ECPDIV bits determine the external clock (ECP clock) frequency as a ratio of VBUS clock or OSCIN as shown in the formula: $ECLK = \frac{VCLK \text{ or } OSCIN}{(ECPDIV + 1)}$ <b>Note: The ECLK pin buffer does not support output frequencies greater than 20 MHz.</b> (1)

### 2.5.1.47 DEV Parity Control Register 1 (DEVCR1)

This register is shown in [Figure 2-52](#) and described in [Table 2-66](#).

**Figure 2-52. DEV Parity Control Register 1 (DEVCR1) [offset = DCh]**



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

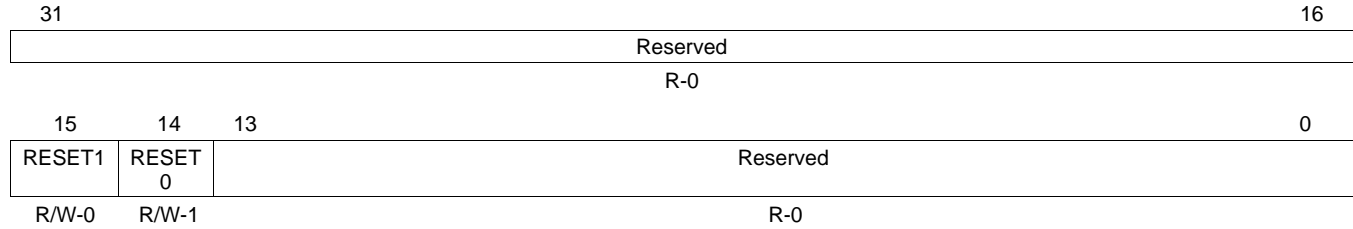
**Table 2-66. DEV Parity Control Register 1 (DEVCR1) Field Descriptions**

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



**2.5.1.48 System Exception Control Register (SYSECR)**

The SYSECR register, shown in [Figure 2-53](#) and described in [Table 2-67](#), is used to generate a software reset.

**Figure 2-53. System Exception Control Register (SYSECR) [offset = E0h]**


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

**Table 2-67. System Exception Control Register (SYSECR) Field Descriptions**

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

### 2.5.1.49 System Exception Status Register (SYSESR)

The SYSESR register, shown in Figure 2-54 and described in Table 2-68, shows the source for different resets encountered. Previous reset source status bits are not automatically cleared if new resets occur. All bits in this register can be cleared by writing a '1' to the bit.

**Figure 2-54. System Exception Status Register (SYSESR) [offset = E4h]**

Reserved										
R-0										
15		14		13		12		8		
PORST		OSCRST		WDRST		Reserved				
R/WC-X		R/WC-X		R/WC-X		R-0				
7			6		5		4		3	
Reserved			CPURST		SWRST		EXTRST		Reserved	
R-0			R/WC-X		R/WC-X		R/WC-X		R-0	
1		0		MPMODE						
R-0		R-0						R-0		

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

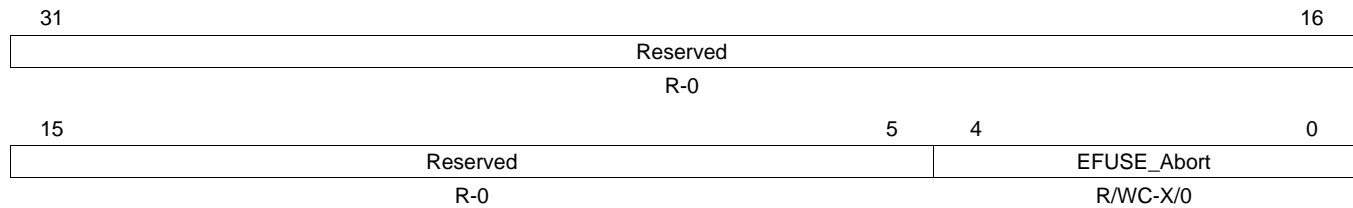
**Table 2-68. System Exception Status Register (SYSESR) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15	PORST	0	Power-up reset. This bit is set when VCCOR (VCC Out of Range) is detected. No reset has occurred due to the V <sub>CC</sub> being out of range.
		1	A reset was caused by V <sub>CC</sub> being out of range (nPORST was asserted).
14	OSCRST	0	Reset caused by an oscillator failure or PLL cycle slip. This bit is set when a reset is caused by an oscillator failure or PLL slip. <b>Note: The action taken when an oscillator failure or PLL slip is detected must be configured in the PLLCTL1 register.</b>
		1	No reset has occurred due to an oscillator failure or a PLL cycle slip. A reset was caused by an oscillator failure or a PLL cycle slip.
13	WDRST	0	Watchdog reset flag. This bit is set when the last reset was caused by the analog or digital watchdog.
		1	No reset has occurred because of the AWD or DWD. A reset was caused by the AWD or DWD.
12-6	Reserved	0	Read returns 0. Writes have no effect.
5	CPURST	0	CPU reset flag. This bit is set when the CPU is reset. <b>Note: A CPU reset can be initiated by the CPU self-test controller (LBIST) or by changing the memory protection (MMU/MPU) configuration in MMUGCR register.</b>
		1	No CPU reset has occurred. A CPU reset occurred.
4	SWRST	0	Software reset flag. This bit is set when a software system reset has occurred. <b>Note: A software system reset can be initiated by writing to the RESET bits in the SYSECR register.</b>
		1	No software reset has occurred. A software reset occurred.
3	EXTRST	0	External reset flag. This bit is set when a reset is caused by the external reset pin nRST. The external reset pin has not asserted a reset.
		1	A reset has been caused by the external reset pin.
2-1	Reserved	0	Read returns 0. Writes have no effect.
0	MPMODE	0	This indicates the current memory protection unit (MPU) mode. MPU is disabled.
		1	MPU is enabled.

### 2.5.1.50 System Test Abort Status Register (SYSTASR)

This register is shown in [Figure 2-55](#) and described in [Table 2-69](#).

**Figure 2-55. System Test Abort Status Register (SYSTASR) [offset = E8h]**



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

**Table 2-69. System Test Abort Status Register (SYSTASR) Field Descriptions**

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

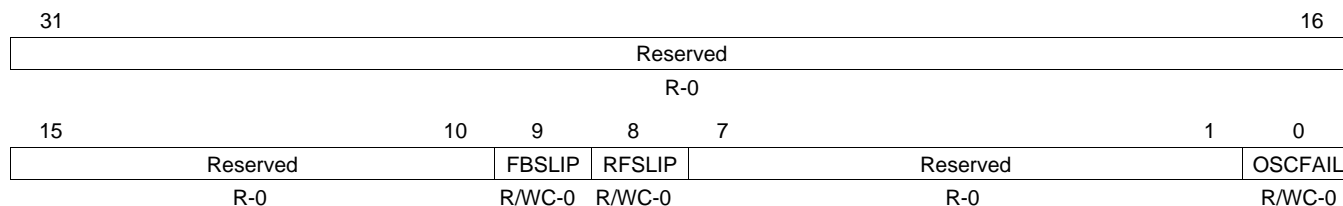
### 2.5.1.51 Global Status Register (GLBSTAT)

The GLBSTAT register, shown in [Figure 2-56](#) and described in [Table 2-70](#), indicates the FMzPLL (PLL1) slip status and the oscillator fail status.

**NOTE: PLL and OSC fail behavior**

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

**Figure 2-56. Global Status Register (GLBSTAT) [offset = ECh]**



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

**Table 2-70. Global Status Register (GLBSTAT) Field Descriptions**

Bit	Field	Value	Description
31-10	Reserved	0	Read returns 0. Writes have no effect.
9	FBSLIP	0	PLL over cycle slip detection. <i>Read:</i> No PLL over cycle slip has been detected. <i>Write:</i> The bit is unchanged.
		1	<i>Read:</i> A PLL over cycle slip has been detected. <i>Write:</i> The bit is cleared to 0.
8	RFSLIP	0	PLL under cycle slip detection. <i>Read:</i> No PLL under cycle slip has been detected. <i>Write:</i> The bit is unchanged.
		1	<i>Read:</i> A PLL under cycle slip has been detected. <i>Write:</i> The bit is cleared to 0.
7-1	Reserved	0	Read returns 0. Writes have no effect.
0	OSCFAIL	0	Oscillator fail flag bit. <i>Read:</i> No oscillator failure has been detected. <i>Write:</i> The bit is unchanged.
		1	<i>Read:</i> An oscillator failure has been detected. <i>Write:</i> The bit is cleared to 0.

### 2.5.1.52 Device Identification Register (DEVID)

The DEVID is a read-only register. It contains device-specific information that is hard-coded during device manufacture. For the initial silicon version, the device identification code value is 802A AD05h. This register is shown in [Figure 2-57](#) and described in [Table 2-71](#).

**Figure 2-57. Device Identification Register (DEVID) [offset = F0h]**

31	30							17	16
CP15	UNIQUE ID						TECH		
R-K	R-K						R-K		
15	13	12	11	10	9	8			
TECH		I/O VOLTAGE	PERIPHERAL PARITY	FLASH ECC		RAM ECC			
R-K		R-K	R-K	R-K		R-K			
7	VERSION			3	2	1	PLATFORM ID		
R-K				R-K					

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

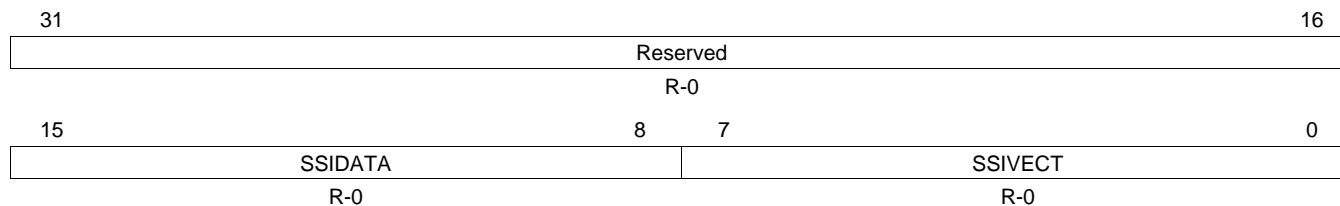
**Table 2-71. Device Identification Register (DEVID) Field Descriptions**

Bit	Field	Value	Description
31	CP15	0 1	CP15 CPU. This bit indicates whether the CPU has a coprocessor 15 (CP15). The CPU has no CP15 present. The CPU has a CP15 present. The CPU ID can be read using the CP15 C0,C0,0 register.
30-17	UNIQUE ID	0-3FFFh	Device ID. The device ID is unique by device configuration.
16-13	TECH	0 1h 2h 3h 4h 5h	These bits define the process technology by which the device was manufactured. Device manufactured in the C05 process technology. Device manufactured in the F05 process technology. Device manufactured in the C035 process technology. Device manufactured in the F035 process technology. Device manufactured in the C021 process technology. Device manufactured in the F021 process technology.
12	I/O VOLTAGE	0 1	Input/output voltage. This bit defines the I/O voltage of the device. The I/O voltage is 3.3 V. The I/O voltage is 5 V.
11	PERIPHERAL PARITY	0 1	Peripheral parity. This bit indicates whether or not peripheral memory parity is present. The peripheral memories have no parity. The peripheral memories have parity.
10-9	FLASH ECC	0 1h 2h 3h	These bits indicate which parity is present for the program memory. No memory protection is present. The program memory (Flash) has single bit parity. The program memory (Flash) has ECC. This combination is reserved.
8	RAM ECC	0 1	RAM ECC. This bit indicates whether or not RAM memory ECC is present. The RAM memories do not have ECC. The RAM memories have ECC.
7-3	VERSION	0-1Fh	Version. These bits provide the revision of the device.
2-0	PLATFORM ID	5h	The device is part of the TMS570Px family. The TMS570Px ID is always 5h.

### 2.5.1.53 Software Interrupt Vector Register (SSIVEC)

The SSIVEC register, shown in [Figure 2-58](#) and described in [Table 2-72](#), contains information about software interrupts.

**Figure 2-58. Software Interrupt Vector Register (SSIVEC) [offset = F4h]**



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

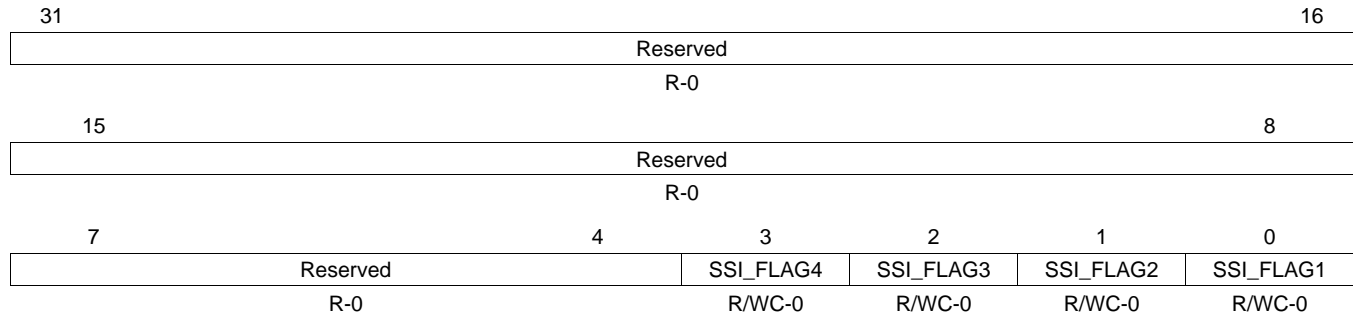
**Table 2-72. Software Interrupt Vector Register (SSIVEC) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-8	SSIDATA	0-FFh	System software interrupt data key. These bits contain the data key value of the source for the system software interrupt, which is indicated by the vector in the SSIVEC[7-0] field.
7-0	SSIVECT		<p>These bits contain the source for the system software interrupt.</p> <p><b>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.</b></p> <p><b>Note: The SSIR [4-1] interrupt has the following priority order:</b>            SSIR1 has the highest priority.            SSIR4 has the lowest priority.</p> <p style="margin-left: 20px;">0      No software interrupt is pending.            1h     A software interrupt has been generated by writing the correct key value to The SSIR1 register.            2h     A software interrupt has been generated by writing the correct key value to The SSIR2 register.            3h     A software interrupt has been generated by writing the correct key value to The SSIR3 register.            4h     A software interrupt has been generated by writing the correct key value to The SSIR4 register.            5h-FFh    Reserved</p>

### 2.5.1.54 System Software Interrupt Flag Register (SSIF)

The SSIF register, shown in Figure 2-59 and described in Table 2-73, contains software interrupt flag status information.

**Figure 2-59. System Software Interrupt Flag Register (SSIF) [offset = F8h]**



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

**Table 2-73. System Software Interrupt Flag Register (SSIF) Field Descriptions**

Bit	Field	Value	Description
31-4	Reserved	0	Read returns 0. Writes have no effect.
3-0	SSI_FLAG[4-1]	0	System software interrupt flag[4-1]. This flag is set when the correct SSKEY is written to the SSIR register[4-1].  <b>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.</b>  <i>Read:</i> No IRQ/FIQ interrupt was generated since the bit was last cleared. <i>Write:</i> The bit is unchanged.
		1	<i>Read:</i> An IRQ/FIQ interrupt was generated. <i>Write:</i> The bit is cleared to 0.



## 2.5.2 Secondary System Control Registers (SYS2)

This section describes the secondary frame of system registers. 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.

**Table 2-74. Secondary System Control Registers Summary**

Offset	Acronym	Register Description	Section
00	PLLCTL3	PLL Control Register 3	<a href="#">Section 2.5.2.1</a>
08h	STCCLKDIV	CPU Logic BIST Clock Divider	<a href="#">Section 2.5.2.2</a>
3Ch	CLK2CNTRL	Clock 2 Control Register	<a href="#">Section 2.5.2.3</a>
40h	VCLKACON1	Peripheral Asynchronous Clock Configuration 1 Register	<a href="#">Section 2.5.2.4</a>
70h	CLKSLIP	Clock Slip Register	<a href="#">Section 2.5.2.5</a>
ECh	EFC_CTLREG	EFUSE Controller Control Register	<a href="#">Section 2.5.2.6</a>
F0h	DIEDL_REG0	Die Identification Register Lower Word	<a href="#">Section 2.5.2.7</a>
F4h	DIEDH_REG1	Die Identification Register Upper Word	<a href="#">Section 2.5.2.8</a>
F8h	DIEDL_REG2	Die Identification Register Lower Word	<a href="#">Section 2.5.2.9</a>
FCh	DIEDH_REG3	Die Identification Register Upper Word	<a href="#">Section 2.5.2.10</a>

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

### 2.5.2.1 PLL Control Register 3 (PLLCTL3)

The PLLCTL3 register is shown in [Figure 2-60](#) and described in [Table 2-75](#); controls the settings of PLL2 (Clock Source 6 - FPLL).

**Figure 2-60. PLL Control Register 3 (PLLCTL3) [offset = 00]**

31	29	28	24	23	22	21	16
ODPLL2		PLLDIV2			Reserved		REFCLKDIV2
R/W-1h		R/W-Fh			R-0		R/W-2h
							0
PLL_MUL2							
R/W-5F00h							

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

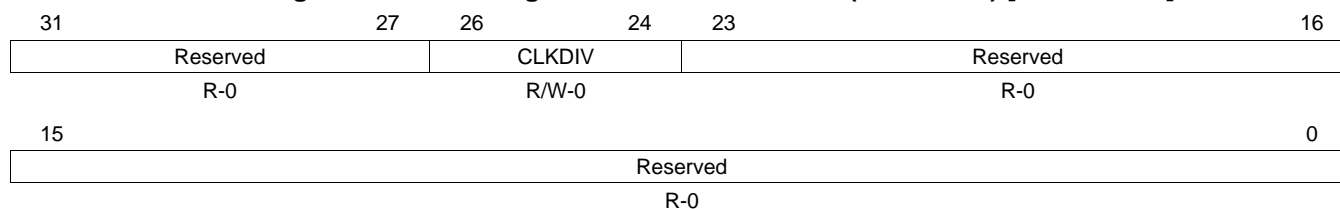
**Table 2-75. PLL Control Register 3 (PLLCTL3) Field Descriptions**

Bit	Field	Value	Description
31-29	ODPLL2	0 to 7	Internal PLL Output Divider, $OD2 = ODPLL2 + 1$ , ranges from 1 to 8. $f_{post\_ODCLK2} = f_{output\_CLK2} / OD2$ Note: PLL output clock is gated off if ODPLL2 is changed while the PLL#2 is active.
28-24	PLLDIV2	0 to 31	PLL#2 Output Clock Divider, $R2 = PLLDIV2 + 1$ , ranges from 1 to 32. $f_{PLL2\_CLK} = f_{post\_ODCLK2} / R2$
23-22	Reserved	0	Read/Write, but value has no effect on PLL operation.
21-16	REFCLKDIV2	0 to 63	Reference Clock Divider, $NR2 = REFCLKDIV2 + 1$ , ranges from 1 to 64. $f_{INTCLK2} = f_{OSCIN} / NR2$ Note: This value should not be changed while the PLL#2 is active.
15-0	PLLMUL2	Valid multiplication factors are from 1 to 256. PLL#2 Multiplier, $NF2 = (PLLMUL2 / 256) + 1$ $f_{VCOCLK2} = f_{INTCLK2} \times NF2$ User and privileged mode (read) Privileged mode (write): 5B00h $f_{VCOCLK2} = f_{INTCLK2} \times 92$ 5C00h $f_{VCOCLK2} = f_{INTCLK2} \times 93$ ... continues in sequence 0B70h $f_{VCOCLK2} = f_{INTCLK2} \times 184$	

### 2.5.2.2 CPU Logic Bist Clock Divider (STCLKDIV)

This register is shown in [Figure 2-61](#) and described in [Table 2-76](#).

**Figure 2-61. CPU Logic BIST Clock Prescaler (STCLKDIV) [offset = 08h]**



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

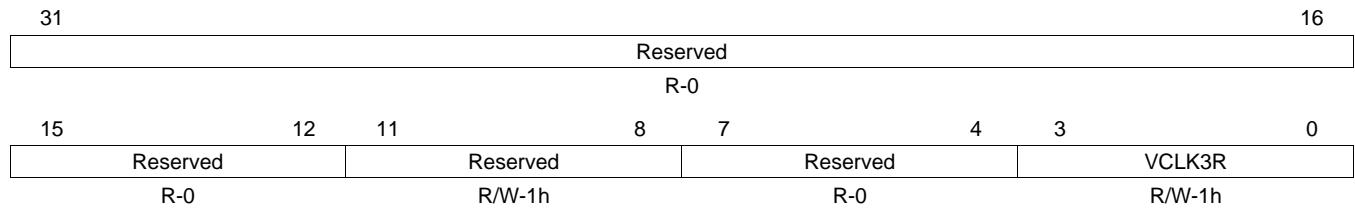
**Table 2-76. CPU Logic BIST Clock Prescaler (STCLKDIV) Field Descriptions**

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

### 2.5.2.3 Clock 2 Control Register (CLK2CNTRL)

This register is shown in [Figure 2-62](#) and described in [Table 2-77](#).

**Figure 2-62. Clock 2 Control Register (CLK2CNTRL) [offset = 3Ch]**



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

**Table 2-77. Clock 2 Control Register (CLK2CNTRL) Field Descriptions**

Bit	Field	Value	Description
31-12	Reserved	0	Read returns 0. Writes have no effect.
11-8	Reserved		Reads return value and writes allowed in privilege mode.
7-4	Reserved	0	Read returns 0. Writes have no effect.
3-0	VCLK3R	0	VBUS clock3 ratio. The ratio is HCLK divide by 1.
		...	...
		Fh	The ratio is HCLK divided by 16.

### 2.5.2.4 Peripheral Asynchronous Clock Configuration 1 Register (VCLKACON1)

This register is shown in [Figure 2-63](#) and described in [Table 2-78](#).

**Figure 2-63. Peripheral Asynchronous Clock Configuration 1 Register (VCLKACON1) [offset = 40h]**

31	27	26	24	23	21	20	19	16
Reserved		VCLKA4R		Reserved		VCLKA4_DIV_C DDIS	VCLKA4S	
R-0		R/W-1h		R-0		R/W-0	R/W- 9h	
15	11	10	8	7	5	4	3	0
Reserved		VCLKA3R		Reserved		VCLKA3_DIV_C DDIS	VCLKA3S	
R-0		R/W-1h		R-0		R/W-0	R/W- 9h	

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

**Table 2-78. Peripheral Asynchronous Clock Configuration 1 Register (VCLKACON1) Field Descriptions**

Bit	Field	Value	Description
31-27	Reserved	0	Read returns 0. Writes have no effect.
26-24	VCLKA4R	0 ... 7h	Clock divider for the VCLKA4 source. Output will be present on VCLKA4_DIVR. VCLKA4 domain will be enabled by writing to the CDDIS register and VCLKA4_DIV_CDDIS bit It can inferred that VCLKA4_DIV clock is disabled when VCLKA4_S clock is disabled. The ratio is VCLKA4 divided by 1. ... The ratio is VCLKA4 divided by 8.
23-21	Reserved	0	Read returns 0. Writes have no effect.
20	VCLKA4_DIV_CDDIS	0 1	Disable the VCLKA4 divider output. VCLKA4 domain will be enabled by writing to the CDDIS register 0 Disable the prescaled VCLKA4 clock on VCLKA4_DIVR. 1 Enable the prescaled VCLKA4 clock on VCLKA4_DIVR.
19-16	VCLKA4S	0 1h 2h 3h 4h 5h 6h 7h 8h-Fh	Peripheral asynchronous clock4 source. 0 Clock source0 is the source for peripheral asynchronous clock4. 1h Clock source1 is the source for peripheral asynchronous clock4. 2h Clock source2 is the source for peripheral asynchronous clock4. 3h Clock source3 is the source for peripheral asynchronous clock4. 4h Clock source4 is the source for peripheral asynchronous clock4. 5h Clock source5 is the source for peripheral asynchronous clock4. 6h Clock source6 is the source for peripheral asynchronous clock4. 7h Clock source7 is the source for peripheral asynchronous clock4. 8h-Fh VCLK is the source for peripheral asynchronous clock4.
15-11	Reserved	0	Read returns 0. Writes have no effect.
10-8	VCLKA3R	0 ... 7h	Clock divider for the VCLKA3 source. Output will be present on VCLKA3_DIVR. VCLKA3 domain will be enabled by writing to the CDDIS register and VCLKA3_DIV_CDDIS bit It can inferred that VCLKA3_DIV clock is disabled when VCLKA3_S clock is disabled. The ratio is VCLKA3 divided by 1. ... The ratio is VCLKA3 divided by 8.
7-5	Reserved	0	Read returns 0. Writes have no effect.

**Table 2-78. Peripheral Asynchronous Clock Configuration 1 Register (VCLKACON1) Field Descriptions  
(continued)**

Bit	Field	Value	Description
4	VCLKA3_DIV_ CDDIS	0 1	Disable the VCLKA3 divider output. VCLKA3 domain will be enabled by writing to the CDDIS register Disable the prescaled VCLKA3 clock on VCLKA3_DIVR. Enable the prescaled VCLKA3 clock on VCLKA3_DIVR.
3-0	VCLKA3S	0 1h 2h 3h 4h 5h 6h 7h 8h-Fh	Peripheral asynchronous clock3 source. Clock source0 is the source for peripheral asynchronous clock3. Clock source1 is the source for peripheral asynchronous clock3. Clock source2 is the source for peripheral asynchronous clock3. Clock source3 is the source for peripheral asynchronous clock3. Clock source4 is the source for peripheral asynchronous clock3. Clock source5 is the source for peripheral asynchronous clock3. Clock source6 is the source for peripheral asynchronous clock3. Clock source7 is the source for peripheral asynchronous clock3. VCLK is the source for peripheral asynchronous clock3.

**Note: Non implemented clock sources should not be enabled or used. A list of the available clock sources is shown in the Clock Sources Table.**

### 2.5.2.5 Clock Slip Register (CLKSLIP)

This register is shown in [Figure 2-64](#) and described in [Table 2-79](#).

**Figure 2-64. Clock Slip Register (CLKSLIP) [offset = 70h]**

31	Reserved												16
R-0													
15	14	13					8	7			4	3	0
Reserved		PLL1_SLIP_FILTER_COUNT				Reserved		PLL1_SLIP_FILTER_KEY					
R-0		R/W-0				R-0		R/W-5h					

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

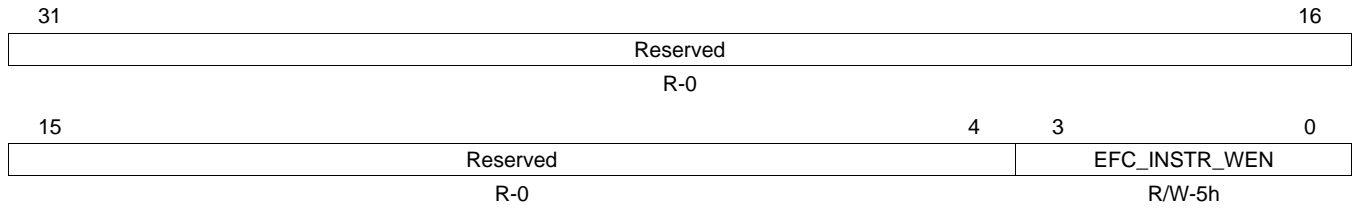
**Table 2-79. Clock Slip Register (CLKSLIP) Field Descriptions**

Bit	Field	Value	Description
31-14	Reserved	0	Read returns 0. Writes have no effect.
13-8	PLL1_SLIP_FILTER_COUNT	0 3Fh - 1h	Configure the count for the filtered PLL slip. Count is on 10M clock. On reset, counter value is zero. Counter must be programmed to non-zero value and enabled for the filtering to be enabled Counter programmed to a non-zero value
7-4	Reserved	0	Read returns 0. Writes have no effect.
3-0	PLL1_SLIP_FILTER_KEY	5h Fh Others	Enable the PLL filtering. On reset, the PLL filter is disabled and normal PLL slip passes through. This is an unsupported value. User should avoid writing this value to this field. Enable of filtered PLL slip. Recommended to program 1010 in these bit fields. Enable of PLL slip occurs when the KEY is programmed and a non-zero value is present in the COUNT field.

**2.5.2.6 EFUSE Controller Control Register (EFC\_CTLREG)**

This register is shown in [Figure 2-65](#) and described in [Table 2-80](#).

**Figure 2-65. EFUSE Controller Control Register (EFC\_CTLREG) [offset = ECh]**



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

**Table 2-80. EFUSE Controller Control Register (EFC\_CTLREG) Field Descriptions**

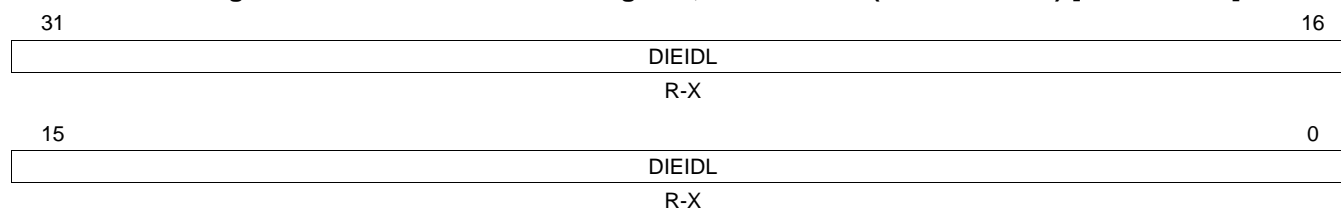
Bit	Field	Value	Description
31-4	Reserved	0	Read returns 0. Writes have no effect.
3-0	EFC_INSTR_WEN	Ah	Enable user write of 4 EFUSE controller instructions. SYS module generates the enable signal which will be tied to OCP_FROM_WRITE_DISABLE on efuse controller port Writing of instructions (Program,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



### 2.5.2.7 Die Identification Register Lower Word (DIEIDL\_REG0)

This register is shown in [Figure 2-66](#) and described in [Table 2-81](#).

**Figure 2-66. Die Identification Register, Lower Word (DIEIDL\_REG0) [offset = F0h]**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset; -X = Value unchanged after reset

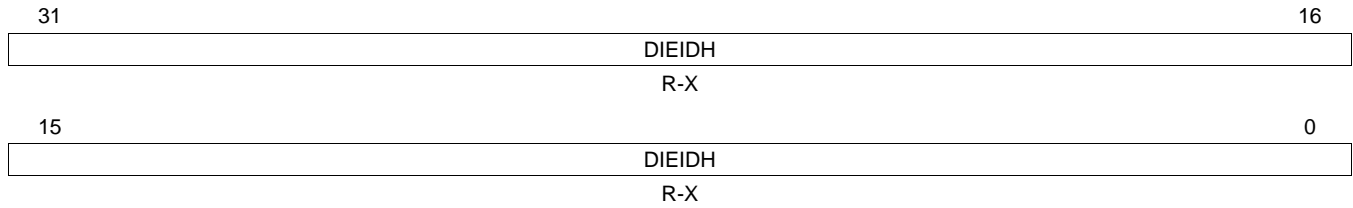
**Table 2-81. Die Identification Register, Lower Word (DIEIDL\_REG0) Field Descriptions**

Bit	Field	Value	Description
31-0	DIEIDL(31-0)	0-FFFF FFFFh	This read-only register contains the lower word (31:0) of the die ID information.

**2.5.2.8 Die Identification Register Upper Word (DIEIDH\_REG1)**

This register is shown in [Figure 2-67](#) and described in [Table 2-82](#).

**Figure 2-67. Die Identification Register, Upper Word (DIEIDH\_REG1) [offset = F4h]**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset; -X = Value unchanged after reset

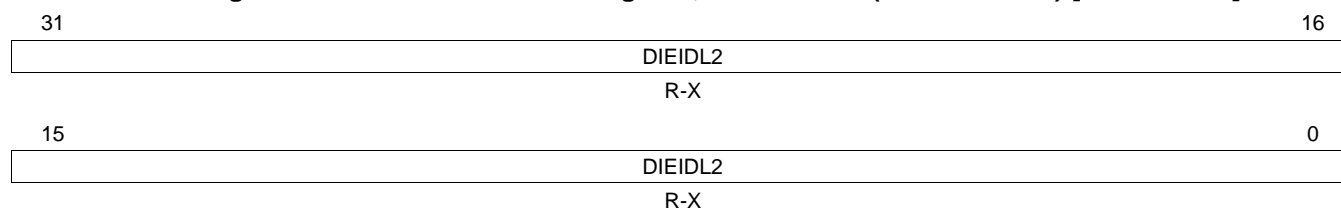
**Table 2-82. Die Identification Register, Upper Word (DIEIDH\_REG1) Field Descriptions**

Bit	Field	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.

### 2.5.2.9 Die Identification Register Lower Word (DIEIDL\_REG2)

This register is shown in [Figure 2-68](#) and described in [Table 2-83](#).

**Figure 2-68. Die Identification Register, Lower Word (DIEIDL\_REG2) [offset = F8h]**



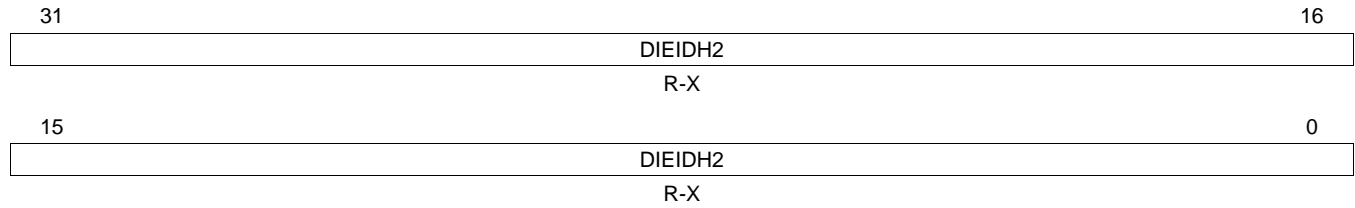
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset; -X = Value unchanged after reset

**Table 2-83. Die Identification Register, Lower Word (DIEIDL\_REG2) Field Descriptions**

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

**2.5.2.10 Die Identification Register Upper Word (DIEIDH\_REG3)**

This register is shown in [Figure 2-69](#) and described in [Table 2-84](#).

**Figure 2-69. Die Identification Register, Upper Word (DIEIDH\_REG3) [offset = FCh]**


LEGEND: R/W = Read/Write; R = Read only; -n = value after reset; -X = Value unchanged after reset

**Table 2-84. Die Identification Register, Upper Word (DIEIDH\_REG3) Field Descriptions**

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

### 2.5.3 Peripheral Central Resource (PCR) Control Registers

This section describes the Peripheral Central Resource (PCR) control registers. The start address of the PCR register frame is 0xFFFF E000 and the end address is 0xFFFF E0FF. [Table 2-85](#) summarizes the registers in the PCR, which are used to configure protection to the peripherals in PCS and PS regions. The following sections provide detailed information about these registers. Not all chip selects exist on this device.

**Table 2-85. Peripheral Central Resource Control Registers Summary**

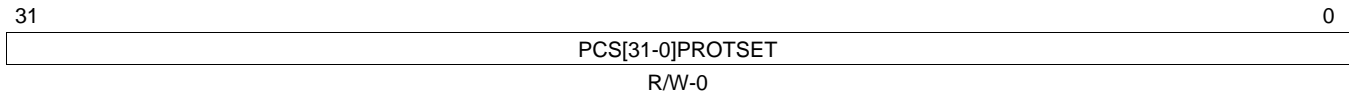
Offset	Acronym	Register Description	Section
00	PMPROTSET0	Peripheral Memory Protection Set Register 0	<a href="#">Section 2.5.3.1</a>
04h	PMPROTSET1	Peripheral Memory Protection Set Register 1	<a href="#">Section 2.5.3.2</a>
10h	PMPROTCLR0	Peripheral Memory Protection Clear Register 0	<a href="#">Section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

### 2.5.3.1 Peripheral Memory Protection Set Register 0 (PMPROTSET0)

This register is shown in [Figure 2-70](#) and described in [Table 2-86](#).

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

**Figure 2-70. Peripheral Memory Protection Set Register 0 (PMPROTSET0) [offset = 00]**



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

**Table 2-86. Peripheral Memory Protection Set Register 0 (PMPROTSET0) Field Descriptions**

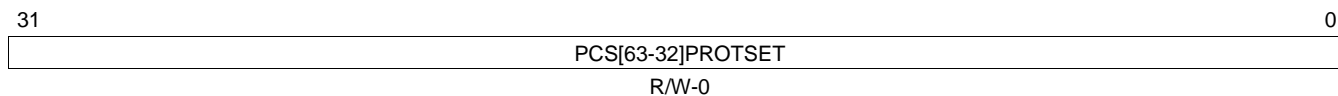
Bit	Field	Value	Description
31-0	PCS[31-0]PROTSET	0	Peripheral memory frame protection set. <i>Read:</i> The peripheral memory frame <i>n</i> can be written to and read from in both user and privileged modes. <i>Write:</i> The bit is unchanged.
		1	<i>Read:</i> The peripheral memory frame <i>n</i> can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

### 2.5.3.2 Peripheral Memory Protection Set Register 1 (PMPROTSET1)

This register is shown in [Figure 2-71](#) and described in [Table 2-87](#).

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

**Figure 2-71. Peripheral Memory Protection Set Register 1 (PMPROTSET1) [offset = 04h]**



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

**Table 2-87. Peripheral Memory Protection Set Register 1 (PMPROTSET1) Field Descriptions**

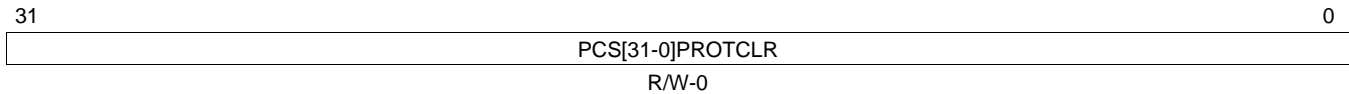
Bit	Field	Value	Description
31-0	PCS[63-32]PROTSET	0	Peripheral memory frame protection set. <i>Read:</i> The peripheral memory frame <i>n</i> can be written to and read from in both user and privileged modes. <i>Write:</i> The bit is unchanged.
		1	<i>Read:</i> The peripheral memory frame <i>n</i> can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

### 2.5.3.3 Peripheral Memory Protection Clear Register 0 (PMPROTCLR0)

This register is shown in [Figure 2-72](#) and described in [Table 2-88](#).

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

**Figure 2-72. Peripheral Memory Protection Clear Register 0 (PMPROTCLR0) [offset = 10h]**



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

**Table 2-88. Peripheral Memory Protection Clear Register 0 (PMPROTCLR0) Field Descriptions**

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

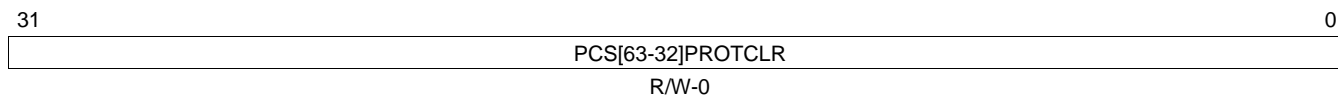


### 2.5.3.4 Peripheral Memory Protection Clear Register 1 (PMPROTCLR1)

This register is shown in [Figure 2-73](#) and described in [Table 2-89](#).

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

**Figure 2-73. Peripheral Memory Protection Clear Register 1 (PMPROTCLR1) [offset = 14h]**



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

**Table 2-89. Peripheral Memory Protection Clear Register 1 (PMPROTCLR1) Field Descriptions**

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

### 2.5.3.5 Peripheral Protection Set Register 0 (PPROTSET0)

There is one bit for each quadrant for PS0 to PS7.

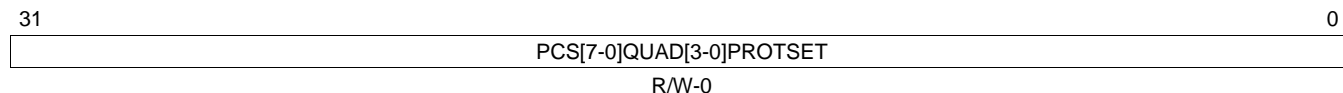
The following are the ways in which quadrants are used within a PS frame-

- (a) The slave uses all the four quadrants  
Only the bit corresponding to the quadrant 0 of PSn is implemented. It protects the whole 1K-byte frame. The remaining three bits are not implemented.
- (b) The slave uses two quadrants  
Each quadrant has to be in one of these groups- (Quad 0 and Quad 1), or (Quad 2 and Quad 3).  
For the group Quad0/Quad1, the bit quadrant 0 protects both quadrants 0 and 1. The bit quadrant 1 is not implemented.  
For the group Quad2/Quad3, the bit quadrant 2 protects both quadrants 2 and 3. The bit quadrant 3 is not implemented.
- (c) The slave uses only one quadrant.  
In this case, the bit, as specified in [Table 2-90](#), protects the slave.

The above arrangement is true for all the peripheral select (PS0 to PS31), presented in [Section 2.5.3.6 - Section 2.5.3.12](#). This register holds bits for PS0 to PS7 and is shown in [Figure 2-74](#) and described in [Table 2-90](#).

**NOTE:** Writes to nonimplemented bits have no effect and reads are 0.

**Figure 2-74. Peripheral Protection Set Register 0 (PPROTSET0) [offset = 20h]**



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

**Table 2-90. Peripheral Protection Set Register 0 (PPROTSET0) Field Descriptions**

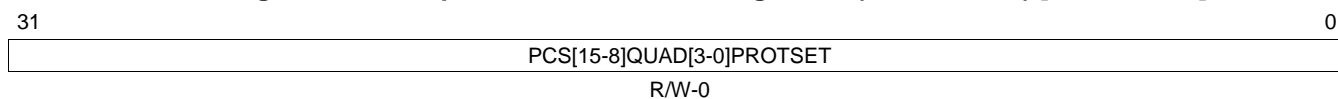
Bit	Field	Value	Description
31-0	PCS[7-0]QUAD[3-0]PROTSET	0	Peripheral select quadrant protection set. <i>Read:</i> The peripheral select quadrant can be written to and read from in both user and privileged modes. <i>Write:</i> The bit is unchanged.
		1	<i>Read:</i> The peripheral select quadrant can be written to only in privileged mode, but it can be read in both user and privileged modes. <i>Write:</i> The corresponding bit in PMPROTSET0 and PMPROTCLR0 registers is set to 1.

### 2.5.3.6 Peripheral Protection Set Register 1 (PPROTSET1)

There is one bit for each quadrant for PS8 to PS15. The protection scheme is described in [Section 2.5.3.5](#). This register is shown in [Figure 2-75](#) and described in [Table 2-91](#).

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

**Figure 2-75. Peripheral Protection Set Register 1 (PPROTSET1) [offset = 24h]**



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

**Table 2-91. Peripheral Protection Set Register 1 (PPROTSET1) Field Descriptions**

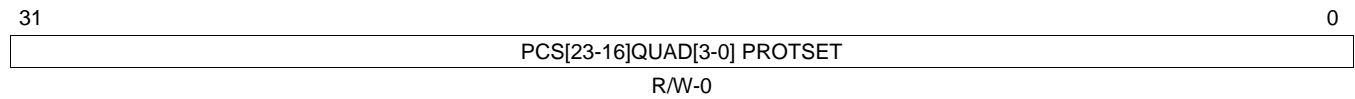
Bit	Field	Value	Description
31-0	PCS[15-8]QUAD[3-0]PROTSET	0	Peripheral select quadrant protection set. <i>Read:</i> The peripheral select quadrant can be written to and read from in both user and privileged modes. <i>Write:</i> The bit is unchanged.
		1	<i>Read:</i> The peripheral select quadrant can be written to only in privileged mode, but it can be read in both user and privileged modes. <i>Write:</i> The corresponding bit in PMPROTSET1 and PMPROTCLR1 registers is set to 1.

### 2.5.3.7 Peripheral Protection Set Register 2 (PPROTSET2)

There is one bit for each quadrant for PS16 to PS23. The protection scheme is described in [Section 2.5.3.5](#). This register is shown in [Figure 2-76](#) and described in [Table 2-92](#).

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

**Figure 2-76. Peripheral Protection Set Register 2 (PPROTSET2) [offset = 28h]**



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

**Table 2-92. Peripheral Protection Set Register 2 (PPROTSET2) Field Descriptions**

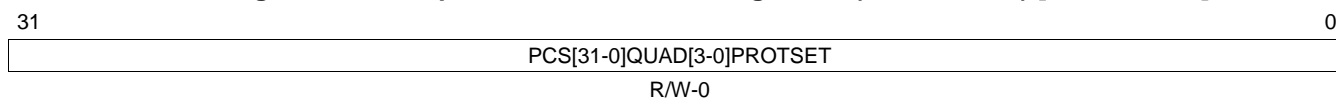
Bit	Field	Value	Description
31-0	PCS[23-16]QUAD[3-0] PROTSET	0	Peripheral select quadrant protection set. <i>Read:</i> The peripheral select quadrant can be written to and read from in both user and privileged modes. <i>Write:</i> The bit is unchanged.
		1	<i>Read:</i> The peripheral select quadrant can be written to only in privileged mode, but it can be read in both user and privileged modes. <i>Write:</i> The corresponding bit in PMPROTSET2 and PMPROTCLR2 registers is set to 1.

### 2.5.3.8 Peripheral Protection Set Register 3 (PPROTSET3)

There is one bit for each quadrant for PS24 to PS31. The protection scheme is described in [Section 2.5.3.5](#). This register is shown in [Figure 2-77](#) and described in [Table 2-93](#).

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

**Figure 2-77. Peripheral Protection Set Register 3 (PPROTSET3) [offset = 2Ch]**



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

**Table 2-93. Peripheral Protection Set Register 3 (PPROTSET3) Field Descriptions**

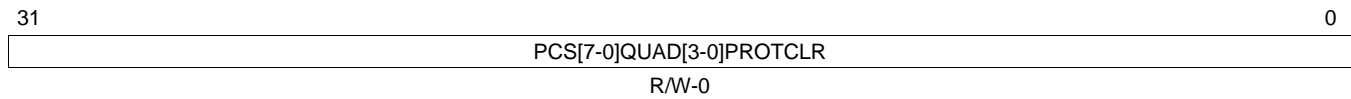
Bit	Field	Value	Description
31-0	PCS[31-0]QUAD[3-0]PROTSET	0	Peripheral select quadrant protection set. <i>Read:</i> The peripheral select quadrant can be written to and read from in both user and privileged modes. <i>Write:</i> The bit is unchanged.
		1	<i>Read:</i> The peripheral select quadrant can be written to only in privileged mode, but it can be read in both user and privileged modes. <i>Write:</i> The corresponding bit in PMPROTSET3 and PMPROTCLR3 registers is set to 1.

### 2.5.3.9 Peripheral Protection Clear Register 0 (PPROTCLR0)

There is one bit for each quadrant for PS0 to PS7. The protection scheme is described in [Section 2.5.3.5](#). This register is shown in [Figure 2-78](#) and described in [Table 2-94](#).

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

**Figure 2-78. Peripheral Protection Clear Register 0 (PPROTCLR0) [offset = 40h]**



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

**Table 2-94. Peripheral Protection Clear Register 0 (PPROTCLR0) Field Descriptions**

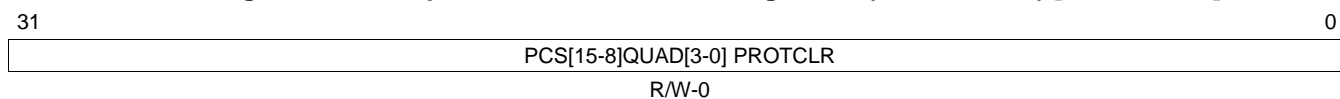
Bit	Field	Value	Description
31-0	PCS[7-0]QUAD[3-0]PROTCLR	0	Peripheral select quadrant protection clear. <i>Read:</i> The peripheral select quadrant can be written to and read from in both user and privileged modes. <i>Write:</i> The bit is unchanged.
		1	<i>Read:</i> The peripheral select quadrant can be written to only in privileged mode, but it can be read in both user and privileged modes. <i>Write:</i> The corresponding bit in PMPROTSET0 and PMPROTCLR0 registers is cleared to 0.

### 2.5.3.10 Peripheral Protection Clear Register 1 (PPROTCLR1)

There is one bit for each quadrant for PS8 to PS15. The protection scheme is described in [Section 2.5.3.5](#). This register is shown in [Figure 2-79](#) and described in [Table 2-95](#).

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

**Figure 2-79. Peripheral Protection Clear Register 1 (PPROTCLR1) [offset = 44h]**



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

**Table 2-95. Peripheral Protection Clear Register 1 (PPROTCLR1) Field Descriptions**

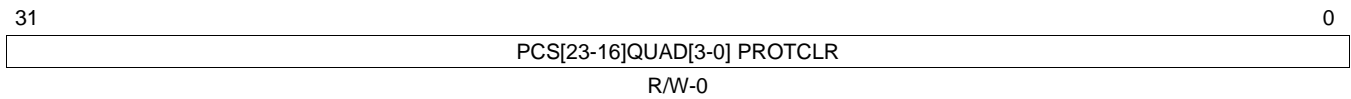
Bit	Field	Value	Description
31-0	PCS[15-8]QUAD[3-0] PROTCCLR	0	Peripheral select quadrant protection clear. <i>Read:</i> The peripheral select quadrant can be written to and read from in both user and privileged modes. <i>Write:</i> The bit is unchanged.
		1	<i>Read:</i> The peripheral select quadrant can be written to only in privileged mode, but it can be read in both user and privileged modes. <i>Write:</i> The corresponding bit in PMPROTSET1 and PMPROTCLR1 registers is cleared to 0.

### 2.5.3.11 Peripheral Protection Clear Register 2 (PPROTCLR2)

There is one bit for each quadrant for PS16 to PS23. The protection scheme is described in [Section 2.5.3.5](#). This register is shown in [Figure 2-80](#) and described in [Table 2-96](#).

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

**Figure 2-80. Peripheral Protection Clear Register 2 (PPROTCLR2) [offset = 48h]**



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

**Table 2-96. Peripheral Protection Clear Register 2 (PPROTCLR2) Field Descriptions**

Bit	Field	Value	Description
31-0	PCS[23-16]QUAD[3-0] PROTCLEAR	0	Peripheral select quadrant protection clear. <i>Read:</i> The peripheral select quadrant can be written to and read from in both user and privileged modes. <i>Write:</i> The bit is unchanged.
		1	<i>Read:</i> The peripheral select quadrant can be written to only in privileged mode, but it can be read in both user and privileged modes. <i>Write:</i> The corresponding bit in PMPROTSET2 and PMPROTCLR2 registers is cleared to 0.

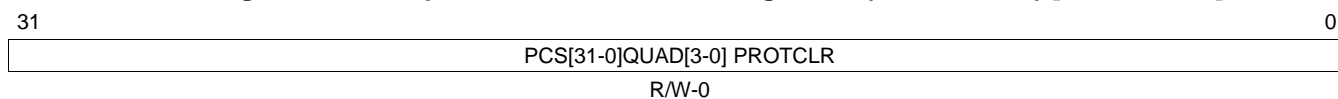


### 2.5.3.12 Peripheral Protection Clear Register 3 (PPROTCLR3)

There is one bit for each quadrant for PS24 to PS31. The protection scheme is described in [Section 2.5.3.5](#). This register is shown in [Figure 2-81](#) and described in [Table 2-97](#).

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

**Figure 2-81. Peripheral Protection Clear Register 3 (PPROTCLR3) [offset = 4Ch]**



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

**Table 2-97. Peripheral Protection Clear Register 3 (PPROTCLR3) Field Descriptions**

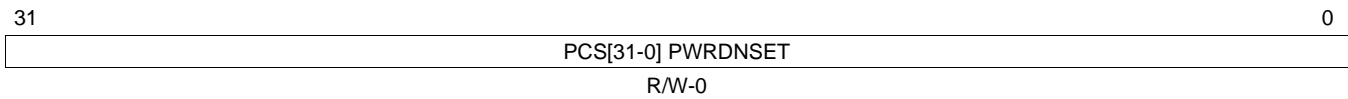
Bit	Field	Value	Description
31-0	PCS[31-0]QUAD[3-0] PROTCLEAR	0	Peripheral select quadrant protection clear. <i>Read:</i> The peripheral select quadrant can be written to and read from in both user and privileged modes. <i>Write:</i> The bit is unchanged.
		1	<i>Read:</i> The peripheral select quadrant can be written to only in privileged mode, but it can be read in both user and privileged modes. <i>Write:</i> The corresponding bit in PMPROTSET3 and PMPROTCLR3 registers is cleared to 0.

### 2.5.3.13 Peripheral Memory Power-Down Set Register 0 (PCSPWRDWNSET0)

Each bit corresponds to a bit at the same index in the PMPROT register in that they both relate to the same peripheral. This register is shown in [Figure 2-82](#) and described in [Table 2-98](#).

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

**Figure 2-82. Peripheral Memory Power-Down Set Register 0 (PCSPWRDWNSET0) [offset = 60h]**



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

**Table 2-98. Peripheral Memory Power-Down Set Register 0 (PCSPWRDWNSET0) Field Descriptions**

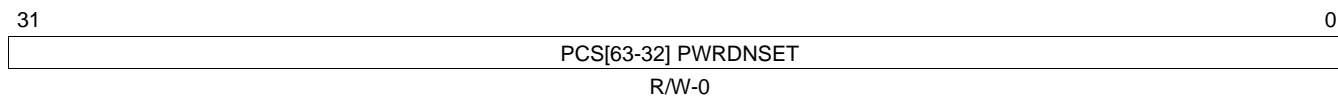
Bit	Field	Value	Description
31-0	PCS[31-0] PWRDNSET	0	Peripheral memory clock power-down set. <i>Read:</i> The peripheral memory clock[31-0] is active. <i>Write:</i> The bit is unchanged.
		1	<i>Read:</i> The peripheral memory clock[31-0] is inactive. <i>Write:</i> The corresponding bit in the PCSPWRDWNSET0 and PCSPWRDWNCLR0 registers is set to 1.

### 2.5.3.14 Peripheral Memory Power-Down Set Register 1 (PCSPWRDWNSET1)

This register is shown in [Figure 2-83](#) and described in [Table 2-99](#).

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

**Figure 2-83. Peripheral Memory Power-Down Set Register 1 (PCSPWRDWNSET1) [offset = 64h]**



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

**Table 2-99. Peripheral Memory Power-Down Set Register 1 (PCSPWRDWNSET1) Field Descriptions**

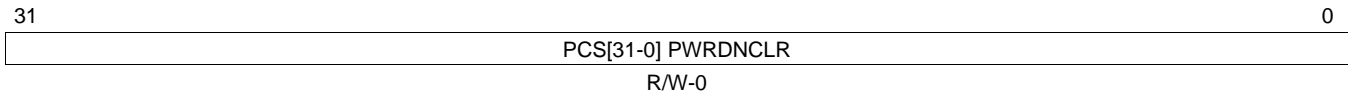
Bit	Field	Value	Description
31-0	PCS[63-32] PWRDNSET	0  1	Peripheral memory clock power-down set. <i>Read:</i> The peripheral memory clock[63-32] is active. <i>Write:</i> The bit is unchanged.  <i>Read:</i> The peripheral memory clock[63-32] is inactive. <i>Write:</i> The corresponding bit in the PCSPWRDWNSET1 and PCSPWRDWNCLR1 registers is set to 1.

### 2.5.3.15 Peripheral Memory Power-Down Clear Register 0 (PCSPWRDWNCLR0)

This register is shown in [Figure 2-84](#) and described in [Table 2-100](#).

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

**Figure 2-84. Peripheral Memory Power-Down Clear Register 0 (PCSPWRDWNCLR0) [offset = 70h]**



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

**Table 2-100. Peripheral Memory Power-Down Clear Register 0 (PCSPWRDWNCLR0) Field Descriptions**

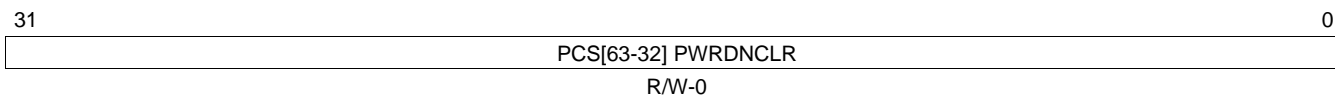
Bit	Field	Value	Description
31-0	PCS[31-0] PWRDNCLR	0	Peripheral memory clock power-down clear. <i>Read:</i> The peripheral memory clock[31-0] is active. <i>Write:</i> The bit is unchanged.
		1	<i>Read:</i> The peripheral memory clock[31-0] is inactive. <i>Write:</i> The corresponding bit in the PCSPWRDWNSET0 and PCSPWRDWNCLR0 registers is cleared to 0.

### 2.5.3.16 Peripheral Memory Power-Down Clear Register 1 (PCSPWRDNCLR1)

This register is shown in [Figure 2-85](#) and described in [Table 2-101](#).

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

**Figure 2-85. Peripheral Memory Power-Down Clear Register 1 (PCSPWRDNCLR1) [offset = 74h]**



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

**Table 2-101. Peripheral Memory Power-Down Set Register 1 (PCSPWRDNCLR1) Field Descriptions**

Bit	Field	Value	Description
31-0	PCS[63-32] PWRDNCLR	0	Peripheral memory clock power-down clear. <i>Read:</i> The peripheral memory clock[63-32] is active. <i>Write:</i> The bit is unchanged.
		1	<i>Read:</i> The peripheral memory clock[63-32] is inactive. <i>Write:</i> The corresponding bit in the PCSPWRDNSET1 and PCSPWRDNCLR1 registers is cleared to 0.

### 2.5.3.17 Peripheral Power-Down Set Register 0 (PSPWRDWNSET0)

There is one bit for each quadrant for PS0 to PS7. Each bit of this register corresponds to the bit at the same index in the corresponding PPROT register in that they relate to the same peripheral. These bits are used to power down/power up the clock to the corresponding peripheral.

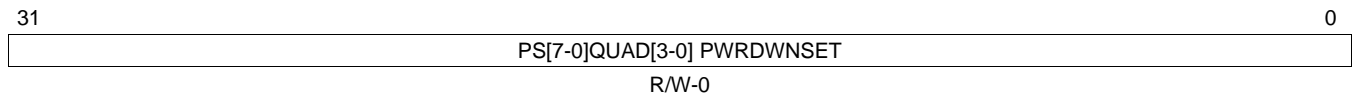
For every bit implemented in the PPROT register, there is one bit in the PSnPWDRWN register, except when two peripherals (both in PS area) share buses. In that case, only one Power-Down bit is implemented, at the position corresponding to that peripheral whose quadrant comes first (the lower numbered).

The ways in which quadrants can be used within a frame are identical to what is described under PPROTSET0, [Section 2.5.3.5](#).

This arrangement is the same for bits of PS8 to PS31, presented in [Section 2.5.3.18](#) - [Section 2.5.3.24](#). This register holds bits for PS0 to PS7. This register is shown in [Figure 2-86](#) and described in [Table 2-102](#).

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

**Figure 2-86. Peripheral Power-Down Set Register 0 (PSPWRDWNSET0) [offset = 80h]**



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

**Table 2-102. Peripheral Power-Down Set Register 0 (PSPWRDWNSET0) Field Descriptions**

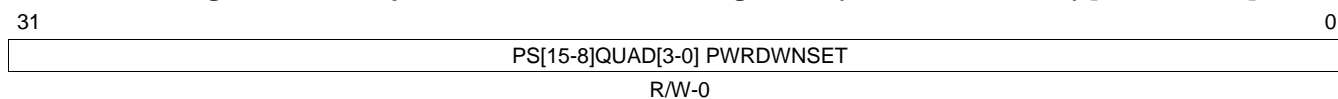
Bit	Field	Value	Description
31-0	PS[7-0]QUAD[3-0] PWRDWNSET	0	Peripheral select quadrant clock power-down set. <i>Read:</i> The clock to the peripheral select quadrant is active. <i>Write:</i> The bit is unchanged.
		1	<i>Read:</i> The clock to the peripheral select quadrant is inactive. <i>Write:</i> The corresponding bit in PSPWRDWNSET0 and PSPWRDWNCLR0 registers is set to 1.

### 2.5.3.18 Peripheral Power-Down Set Register 1 (PSPWRDWNSET1)

There is one bit for each quadrant for PS8 to PS15. The protection scheme is described in [Section 2.5.3.17](#). This register is shown in [Figure 2-87](#) and described in [Table 2-103](#).

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

**Figure 2-87. Peripheral Power-Down Set Register 1 (PSPWRDWNSET1) [offset = 84h]**



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

**Table 2-103. Peripheral Power-Down Set Register 1 (PSPWRDWNSET1) Field Descriptions**

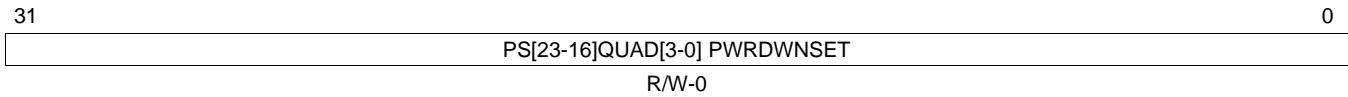
Bit	Field	Value	Description
31-0	PS[15-8]QUAD[3-0] PWRDWNSET	0	Peripheral select quadrant clock power-down set. <i>Read:</i> The clock to the peripheral select quadrant is active. <i>Write:</i> The bit is unchanged.
		1	<i>Read:</i> The clock to the peripheral select quadrant is inactive. <i>Write:</i> The corresponding bit in PSPWRDWNSET1 and PSPWRDWNCLR1 registers is set to 1.

### 2.5.3.19 Peripheral Power-Down Set Register 2 (PSPWRDWNSET2)

There is one bit for each quadrant for PS16 to PS23. The protection scheme is described in [Section 2.5.3.17](#). This register is shown in [Figure 2-88](#) and described in [Table 2-104](#).

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

**Figure 2-88. Peripheral Power-Down Set Register 2 (PSPWRDWNSET2) [offset = 88h]**



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

**Table 2-104. Peripheral Power-Down Set Register 2 (PSPWRDWNSET2) Field Descriptions**

Bit	Field	Value	Description
31-0	PS[23-16]QUAD[3-0] PWRDWNSET	0	Peripheral select quadrant clock power-down set. <i>Read:</i> The clock to the peripheral select quadrant is active. <i>Write:</i> The bit is unchanged.
		1	<i>Read:</i> The clock to the peripheral select quadrant is inactive. <i>Write:</i> The corresponding bit in PSPWRDWNSET2 and PSPWRDWNCLR2 registers is set to 1.

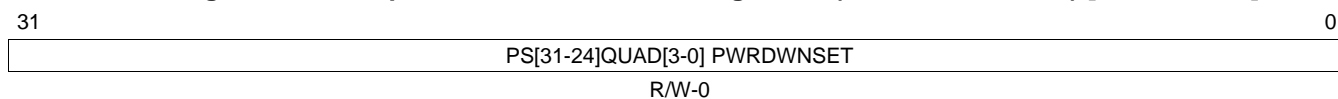


### 2.5.3.20 Peripheral Power-Down Set Register 3 (PSPWRDWNSET3)

There is one bit for each quadrant for PS24 to PS31. The protection scheme is described in [Section 2.5.3.17](#). This register is shown in [Figure 2-89](#) and described in [Table 2-105](#).

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

**Figure 2-89. Peripheral Power-Down Set Register 3 (PSPWRDWNSET3) [offset = 8Ch]**



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

**Table 2-105. Peripheral Power-Down Set Register 3 (PSPWRDWNSET3) Field Descriptions**

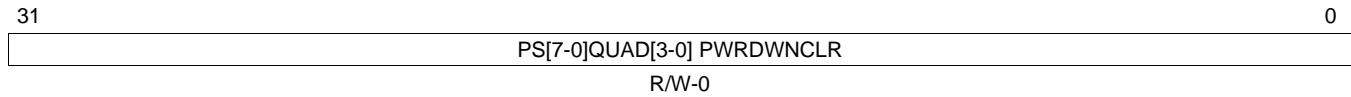
Bit	Field	Value	Description
31-0	PS[31-24]QUAD[3-0] PWRDWNSET	0	Peripheral select quadrant clock power-down set. <i>Read:</i> The clock to the peripheral select quadrant is active. <i>Write:</i> The bit is unchanged.
		1	<i>Read:</i> The clock to the peripheral select quadrant is inactive. <i>Write:</i> The corresponding bit in PSPWRDWNSET3 and PSPWRDWNCLR3 registers is set to 1.

### 2.5.3.21 Peripheral Power-Down Clear Register 0 (PSPWRDWNCLR0)

There is one bit for each quadrant for PS0 to PS7. The protection scheme is described in [Section 2.5.3.17](#). This register is shown in [Figure 2-90](#) and described in [Table 2-106](#).

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

**Figure 2-90. Peripheral Power-Down Clear Register 0 (PSPWRDWNCLR0) [offset = A0h]**



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

**Table 2-106. Peripheral Power-Down Clear Register 0 (PSPWRDWNCLR0) Field Descriptions**

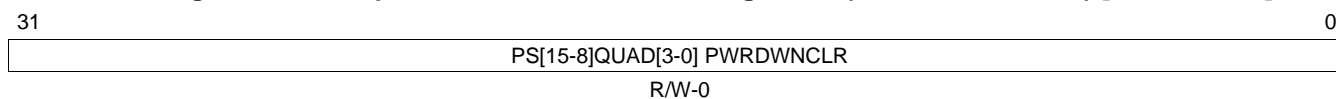
Bit	Field	Value	Description
31-0	PS[7-0]QUAD[3-0] PWRDWNCLR	0	Peripheral select quadrant clock power-down clear. <i>Read:</i> The clock to the peripheral select quadrant is active. <i>Write:</i> The bit is unchanged.
		1	<i>Read:</i> The clock to the peripheral select quadrant is inactive. <i>Write:</i> The corresponding bit in PSPWRDWNSET0 and PSPWRDWNCLR0 registers is cleared to 0.

### 2.5.3.22 Peripheral Power-Down Clear Register 1 (PSPWRDNCLR1)

There is one bit for each quadrant for PS8 to PS15. The protection scheme is described in [Section 2.5.3.17](#). This register is shown in [Figure 2-91](#) and described in [Table 2-107](#).

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

**Figure 2-91. Peripheral Power-Down Clear Register 1 (PSPWRDNCLR1) [offset = A4h]**



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

**Table 2-107. Peripheral Power-Down Clear Register 1 (PSPWRDNCLR1) Field Descriptions**

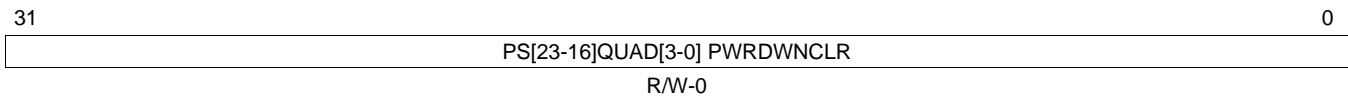
Bit	Field	Value	Description
31-0	PS[15-8]QUAD[3-0] PWRDNCLR	0	Peripheral select quadrant clock power-down clear. <i>Read:</i> The clock to the peripheral select quadrant is active. <i>Write:</i> The bit is unchanged.
		1	<i>Read:</i> The clock to the peripheral select quadrant is inactive. <i>Write:</i> The corresponding bit in PSPWRDNSET1 and PSPWRDNCLR1 registers is cleared to 0.

### 2.5.3.23 Peripheral Power-Down Clear Register 2 (PSPWRDWNCLR2)

There is one bit for each quadrant for PS16 to PS23. The protection scheme is described in [Section 2.5.3.17](#). This register is shown in [Figure 2-92](#) and described in [Table 2-108](#).

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

**Figure 2-92. Peripheral Power-Down Clear Register 2 (PSPWRDWNCLR2) [offset = A8h]**



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

**Table 2-108. Peripheral Power-Down Clear Register 2 (PSPWRDWNCLR2) Field Descriptions**

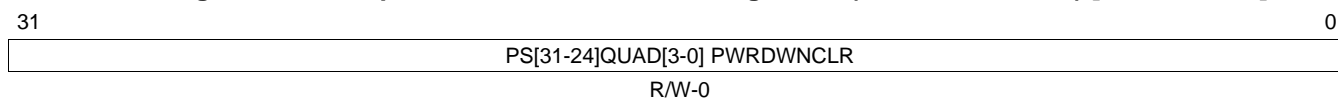
Bit	Field	Value	Description
31-0	PS[23-16]QUAD[3-0] PWRDWNCLR	0	Peripheral select quadrant clock power-down clear. <i>Read:</i> The clock to the peripheral select quadrant is active. <i>Write:</i> The bit is unchanged.
		1	<i>Read:</i> The clock to the peripheral select quadrant is inactive. <i>Write:</i> The corresponding bit in PSPWRDWNSET2 and PSPWRDWNCLR2 registers is cleared to 0.

### 2.5.3.24 Peripheral Power-Down Clear Register 3 (PSPWRDNCLR3)

There is one bit for each quadrant for PS24 to PS31. The protection scheme is described in [Section 2.5.3.17](#). This register is shown in [Figure 2-93](#) and described in [Table 2-109](#).

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

**Figure 2-93. Peripheral Power-Down Clear Register 3 (PSPWRDNCLR) [offset = ACh]**



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

**Table 2-109. Peripheral Power-Down Clear Register 3 (PSPWRDNCLR3) Field Descriptions**

Bit	Field	Value	Description
31-0	PS[31-24]QUAD[3-0] PWRDNCLR	0	Peripheral select quadrant clock power-down clear. <i>Read:</i> The clock to the peripheral select quadrant is active. <i>Write:</i> The bit is unchanged.
		1	<i>Read:</i> The clock to the peripheral select quadrant is inactive. <i>Write:</i> The corresponding bit in PSPWRDNSET3 and PSPWRDNCLR3 registers is cleared to 0.

## ***Power Management Module (PMM)***

---

---

---

This chapter describes the power management module (PMM).

<b>Topic</b>	<b>Page</b>
<b>3.1 Overview .....</b>	<b>220</b>
<b>3.2 Power Domains .....</b>	<b>222</b>
<b>3.3 PMM Operation .....</b>	<b>223</b>
<b>3.4 PMM Registers .....</b>	<b>226</b>

### 3.1 Overview

The microcontroller is part of the Hercules family of microcontrollers from Texas Instruments for safety-critical applications. Several functions are implemented on this microcontroller targeted towards varied applications. The core logic is divided into several domains that can be independently turned on or off based on the application's requirements. This allows an application to reduce the leakage current for a core domain that has modules that are not being used by the application.

This chapter describes the Power Management Module (PMM). The PMM provides memory-mapped registers that control the states of the supported power domains. The PMM includes interfaces to the Power Mode Controller (PMC) and the Power State Controller (PSCON). The PMC and PSCON control the power up/down sequence of each power domain.

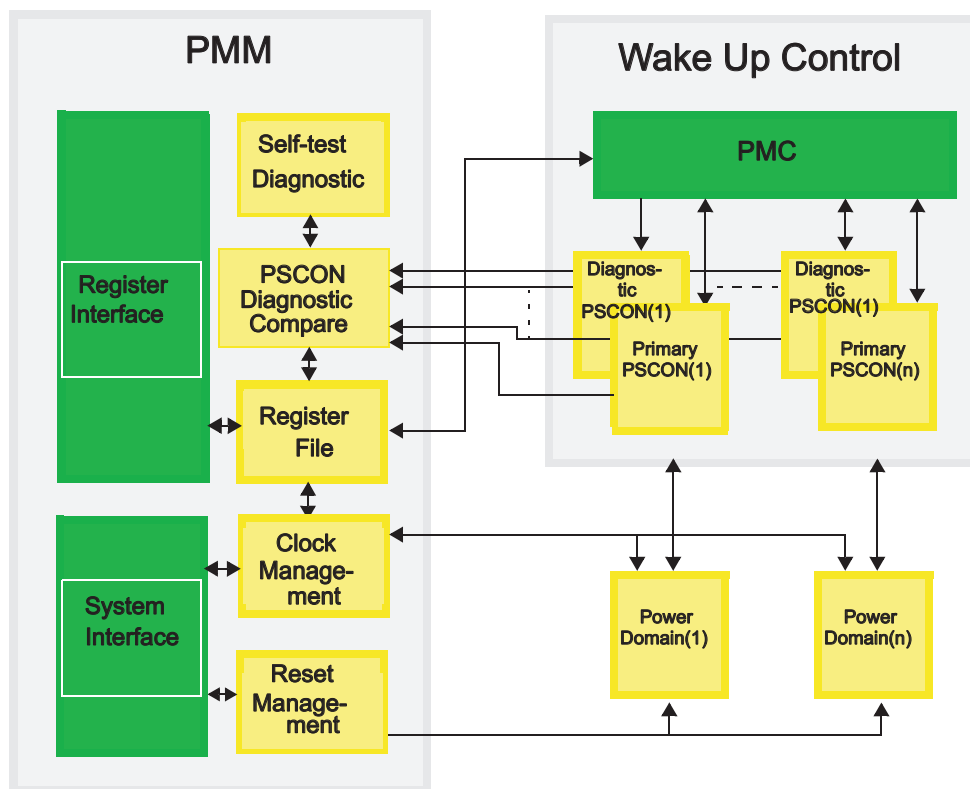
#### 3.1.1 Main Features of the Power Management Module (PMM)

The main features of the PMM implemented on the microcontroller are:

- Supports 5 logic power domains: PD1, PD2, PD3, PD4 and PD5
- Supports 3 memory-only power domains: RAM\_PD1, RAM\_PD2 and RAM\_PD3
- Allows configurable default states for each power domain
- Allows each power domain to be permanently disabled
- Manages the clocks for each power domain
- Manages the resets to each power domain
- Includes failsafe compare logic to continuously monitor the states of each power domain
- Supports diagnostic and self-test logic to validate failsafe compare logic

#### 3.1.2 Block Diagram

Figure 3-1. PMM Block Diagram



PMM consists of several key components:

- Register interface – the PMM control registers are mapped to the device memory space and start at address 0xFFFF0000.
- System Interface – the PMM receives the clocks, resets, errors and all other control signals through this interface.
- PSCON Diagnostic Compare – this block compares the outputs of each primary PSCON and the respective diagnostic PSCON implemented for failsafe safety.
- Self-Test Diagnostic – this block contains the logic to place the PSCON diagnostic compare block in a self-test mode in order to test the failsafe feature.
- Clock management – the PMM provides independent clock gating and handshaking controls for each power domain and also generates the clock domains for each power domain.
- Reset Management – the PMM provides independent reset signals for each power domain.
- Power Mode Controller (PMC) – The PMC is a finite state machine that controls the power sequence from one power mode to another. A power mode is the states of all power domains at a given time. The PMC controls each PSCON to initiate power-up or power-down sequence.
- Power State Controller (PSCON) – The PSCON is a finite state machine that controls the power sequence of a power domain from one state to another. Each power domain is controlled by one dedicated PSCON.
- Power Domain – A power domain is a group of logic and/or memories which is separated from the global power supply via power switches. These power switches are controlled by the PSCON and can be turned on or off.



### 3.2 Power Domains

Figure 3-2. Core Power Domains

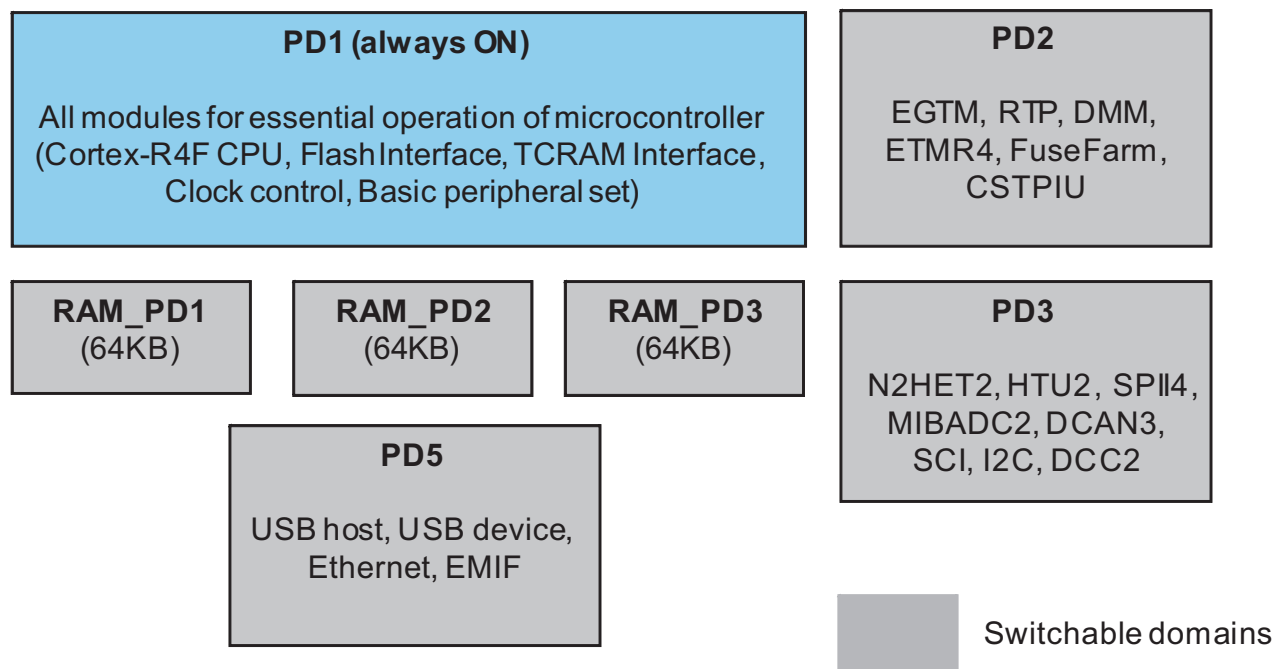


Figure 3-2 shows the core and memory power domains implemented on the microcontroller.

This device has 8 separate core power domains:

- PD1 is an always-ON domain and is not controlled by PMM. It contains the CPU as well as other principal modules and the interconnect required for operation of the microcontroller. This domain also includes 64KB of the tightly-coupled RAM. The PD1 can operate on its own even when all the other core power domains are turned off by the PMM. Note that all I/Os are in this always-ON domain as well.

Core power domains PD2 through PD5 and RAM\_PD1 through RAM\_PD3 are controlled by the PMM.

- PD2 contains the Embedded Trace Macrocell (ETM-R4), RAM Trace Port (RTP), and Data Modification Module (DMM) components of the debug sub-system as well as the Parameter Overlay Module (POM).
- PD3 contains some additional peripheral modules as an enhanced configuration over and above the peripheral set available in PD1. These include a second High-End Timer (NHET2) with its dedicated transfer unit (HTU2), a second Analog-to-Digital Converter (ADC2), a Serial Communication Interface (SCI), an Inter-Integrated Circuit controller (I2C), a third Controller Area Network controller (DCAN3), and a fourth Serial Peripheral Interface module (SPI4).
- PD5 contains the Ethernet controller (EMAC), the External Memory Interface (EMIF), as well as some components of the interconnect fabric required by these modules.
- RAM\_PD1, RAM\_PD2 and RAM\_PD3 each contain 64KB of tightly-coupled RAM.

---

**NOTE: Switching of Power Domains**

The microcontrollers only support static switching of the power domains. That is, the power domains can be turned ON or OFF one time during device initialization. Once configured, it is not allowed to change the state of a power domain without first asserting a system reset.

---

### 3.3 PMM Operation

It is important to understand some fundamental concepts beforehand.

#### 3.3.1 Power Switch

A power domain gets its power supply via a power switch. The power switch creates a link between the global core supply plane and the local switchable power domain supply. Each power domain uses multiple power switches, which are daisy-chained together.

#### 3.3.2 Power Domain State

Each core power domain can be in one of three states: Active, Idle, or Off.

In the **active** state, a power domain is fully powered with normal supply voltage.

In the **idle** state, all clocks to a power domain are turned off (driven low). The supply voltage is still maintained at the normal level.

In the **off** state, a power domain is completely cut off from the power supply.

#### 3.3.3 Default Power Domain State

The default state of each power domain, except for PD1, is controlled by TI during production testing via programming of individual bits within the reset configuration word in the TI-OTP sector of flash bank 0. This allows each power domain to default to either the active state or the off state.

#### 3.3.4 Disabling a Power Domain Permanently

TI can also permanently disable any power domain, except for PD1. This is also controlled by programming of individual bits within the reset configuration word in the TI-OTP sector of flash bank 0.

#### 3.3.5 Changing Power Domain State

A domain can only change state when commanded by the application. Each domain has an associated 4-bit key to define the intended power state. When the correct key is programmed, the PMM initiates the sequence to transition that domain to the commanded state.

A power state transition is considered complete only when every single power switch for that domain has switched over to the commanded state.

##### 3.3.5.1 Turning a Power Domain Off

It is necessary to turn off all clocks going to a power domain before that domain can be powered down. PMM contains the hardware interlocks to handle this. Each power domain has an associated memory-mapped register which allows the application to turn off clocks to that power domain.

Steps to power down a domain with logic – PD2, PD3, PD4, PD5:

1. Write to the PDCLK\_DISx register to disable all clocks to the power domain.
2. Write zeros to the LOGICPDONx register to power down the domain.
3. Poll for LOGICPDPWRSTATx to become “00”. The power domain is now powered down.

A power domain with only SRAM macros does not have a clock input, so the sequence is shorter. This applies to RAM\_PD1, RAM\_PD2 and RAM\_PD3 power domains shown in [Figure 3-2](#):

1. Write the correct key to the MEMPDONx register to power down the domain.
2. Poll for MEMPDPWRSTATx to become “00”. The power domain is now powered down.

### 3.3.5.2 Turning a Power Domain On

A power domain can be turned on by writing the correct key to the LOGICPDON or MEMPDON register. PMM will automatically restart the clocks to the power domain once the power is restored if the “automatic clock enable upon wake up” option is selected. If this option is not selected, the application can turn on clocks to the power domain by clearing the PDCLK\_DIS register. The application must poll the DOMAINISON register to ensure that the power has been fully restored before enabling the clocks.

### 3.3.6 Reset Management

PMM handles the reset sequence for each power domain. When a power domain is turned on from an off state, the PMM will reset the power domain to ensure that all logic begins in its default reset state.

PMM generates nPORRST (power-on reset), nRST (system reset), nPRST (peripheral reset), and nTRST (test / debug logic reset) for each domain.

### 3.3.7 Diagnostic Power State Controller (PSCON)

Each power domain state is controlled by a primary PSCON. There is a second PSCON as well for each power domain. This is the diagnostic PSCON. All power management inputs to a power domain are controlled only by the primary PSCON. All power management outputs from the power domain are fed back to both the primary and the diagnostic PSCON.

The PMM commands both the PSCON identically so that they are always in a lock-step operating mode. A dedicated compare unit checks the outputs of the two PSCON modules on every cycle.

### 3.3.8 PSCON Compare Block

The diagnostic compare block can operate in one of four modes:

#### 3.3.8.1 Lock-Step Mode

This is the default mode of operation of the PSCON compare block. The PSCON diagnostic compare block compares the outputs from the two PSCONs on every cycle. Any mismatch in the PSCON outputs is indicated as a PSCON compare error. This error signal is mapped to the Error Signaling Module’s (ESM) Group1 channel 38. The application can define the response to this error.

#### 3.3.8.2 Self-Test Mode

A self-test mechanism is provided to check the PSCON compare logic for faults. The compare error signal output is disabled in self-test mode. The PSCON diagnostic compare block generates two types of patterns during self-test mode: compare match test followed by compare mismatch test. During the self-test, each test pattern is applied on both PSCON signal ports of the PSCON diagnostic compare block and then is clocked for one cycle. The duration of the self-test is 24 cycles. Any detected fault is indicated as a self-test error, mapped to ESM group1 channel 39. If no fault is detected, the self-test complete flag is set.

The application can poll for this flag to be set and then switch the mode of the PSCON compare block back to lock-step mode by writing to the mode key register.

---

**NOTE: PSCON operation when compare block is in self-test mode**

When the PSCON compare block is in its self-test mode, both PSCONs continue to function normally. However, there is no comparison done on the PSCON outputs.

---

#### Compare match test:

An identical vector is applied to both input ports at the same time, thereby expecting a compare match. If the compare unit produces a mismatch then the self-test error flag is set and the self-test error signal is generated. The compare match test is terminated if a compare mismatch is detected. The compare match test takes 4 cycles to complete when the test passes.

#### Compare mismatch test:

A vector with all 1's is applied to the PSCON diagnostic compare block's primary input port and the same input is also applied to the secondary input port but with one bit flipped starting from bit position 0. The unequal vectors should cause the PSCON diagnostic compare block to generate a compare mismatch at bit position 0. In case a mismatch is not detected, a self-test error is indicated. This compare mismatch test algorithm is repeated until every single bit position is verified on both PSCON signal ports.

### 3.3.8.3 Error-Forcing Mode

This mode is designed specifically to ensure that the error signal output from the PSCON compare block is not stuck inactive. In this mode, a test pattern is applied to the PSCON related inputs of the compare logic to force an error. The application can clear the flag for ESM group1 channel 38 once the error is flagged. If the ESM group1 channel 38 flag does not get set, this indicates that the PSCON compare error signal is stuck inactive and cannot be relied upon to detect a PSCON mismatch.

### 3.3.8.4 Self-Test Error-Forcing Mode

In this mode, an error is forced so that the self-test error output from the PSCON compare block is activated. The application can clear the flag for ESM group1 channel 39 once the error is flagged. If the ESM group1 channel 39 flag does not get set, this indicates that the PSCON compare block self-test error signal is stuck inactive and there is no self-test mechanism available for the PSCON compare block.

### 3.3.8.5 PMM Operation During CPU Halt Debug Mode

No compare errors are generated when the CPU is halted in debug mode, regardless of the mode of the diagnostic compare block. No status flags are updated in this mode. Normal operation of the compare block is resumed once the CPU exits the debug mode.

### 3.4 PMM Registers

[Table 3-1](#) shows a summary of the control registers in the PMM module. The registers support 32-bit, 16-bit and 8-bit accesses. The address offset is specified from the base address of FFFF 0000h. Any access to an unimplemented location within the PMM register frame will generate a bus error that results in an Abort exception.

**Table 3-1. PMM Register Summary**

Offset	Acronym	Register Description	Section
00	LOGICPDPWRCTRL0	Logic Power Domain Control Register 0	<a href="#">Section 3.4.1</a>
10h	MEMPDPWRCTRL0	Memory Power Domain Control Register 0	<a href="#">Section 3.4.2</a>
20h	PDCLKDISREG	Power Domain Clock Disable Register	<a href="#">Section 3.4.3</a>
24h	PDCLKDISSETREG	Power Domain Clock Disable Set Register	<a href="#">Section 3.4.4</a>
28h	PDCLKDISCLRREG	Power Domain Clock Disable Clear Register	<a href="#">Section 3.4.5</a>
40h	LOGICPDPWRSTAT0	Logic Power Domain PD2 Power Status Register	<a href="#">Section 3.4.6</a>
44h	LOGICPDPWRSTAT1	Logic Power Domain PD3 Power Status Register	<a href="#">Section 3.4.7</a>
48h	LOGICPDPWRSTAT2	Logic Power Domain PD4 Power Status Register	<a href="#">Section 3.4.8</a>
4Ch	LOGICPDPWRSTAT3	Logic Power Domain PD5 Power Status Register	<a href="#">Section 3.4.9</a>
80h	MEMPDPWRSTAT0	Memory Power Domain RAM_PD1 Power Status Register	<a href="#">Section 3.4.10</a>
84h	MEMPDPWRSTAT1	Memory Power Domain RAM_PD2 Power Status Register	<a href="#">Section 3.4.11</a>
88h	MEMPDPWRSTAT2	Memory Power Domain RAM_PD3 Power Status Register	<a href="#">Section 3.4.12</a>
A0h	GLOBALCTRL1	Global Control Register 1	<a href="#">Section 3.4.13</a>
A8h	GLOBALSTAT	Global Status Register	<a href="#">Section 3.4.14</a>
ACh	PRCKEYREG	PSCON Diagnostic Compare Key Register	<a href="#">Section 3.4.15</a>
B0h	LPDDCSTAT1	LogicPD PSCON Diagnostic Compare Status Register 1	<a href="#">Section 3.4.16</a>
B4h	LPDDCSTAT2	LogicPD PSCON Diagnostic Compare Status Register 2	<a href="#">Section 3.4.17</a>
B8h	MPDDCSTAT1	Memory PD PSCON Diagnostic Compare Status Register 1	<a href="#">Section 3.4.18</a>
BCh	MPDDCSTAT2	Memory PD PSCON Diagnostic Compare Status Register 2	<a href="#">Section 3.4.19</a>
C0h	ISODIAGSTAT	Isolation Diagnostic Status Register	<a href="#">Section 3.4.20</a>

### 3.4.1 Logic Power Domain Control Register (LOGICPDPWCTRL0)

**Figure 3-3. Logic Power Domain Control Register (LOGICPDPWCTRL0) [offset = 00]**

31	28	27	24	23	20	19	16
Reserved		LOGICPDON0		Reserved		LOGICPDON1	
R-0		R/W-n		R-0		R/W-n	
15	12	11	8	7	4	3	0
Reserved		LOGICPDON2		Reserved		LOGICPDON3	
R-0		R/W-n		R-0		R/W-n	

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

The default values of the control fields are determined by the device reset configuration word stored in the TI-OTP region of flash bank 0.

**Table 3-2. Logic Power Domain Control Register (LOGICPDPWCTRL0) Field Descriptions**

Bit	Field	Value	Description
31-28	Reserved	0	Read returns 0. Writes have no effect.
27-24	LOGICPDON0	Ah 9h Any other value	Read in User and Privileged Mode: Write in Privileged Mode only: Power domain PD2 is in OFF state. Power domain PD2 is commanded to switch to OFF state. Reserved Power domain PD2 is in Active state. Power domain PD2 is commanded to switch to Active state.
23-20	Reserved	0	Read returns 0. Writes have no effect.
19-16	LOGICPDON1	Ah 9h Any other value	Read in User and Privileged Mode: Write in Privileged Mode only: Power domain PD3 is in OFF state. Power domain PD3 is commanded to switch to OFF state. Reserved Power domain PD3 is in Active state. Power domain PD3 is commanded to switch to Active state.
15-12	Reserved	0	Read returns 0. Writes have no effect.
11-8	LOGICPDON2	Ah 9h Any other value	Read in User and Privileged Mode: Write in Privileged Mode only: Power domain PD4 is in OFF state. Power domain PD4 is commanded to switch to OFF state. Reserved Power domain PD4 is in Active state. Power domain PD4 is commanded to switch to Active state.
7-4	Reserved	0	Read returns 0. Writes have no effect.

**Table 3-2. Logic Power Domain Control Register (LOGICPDPWRCTRL0) Field Descriptions (continued)**

Bit	Field	Value	Description
3-0	LOGICPDON3	Ah	Read in User and Privileged Mode: Write in Privileged Mode only: Power domain PD5 is in OFF state. Power domain PD5 is commanded to switch to OFF state.
		9h	Reserved
		Any other value	Power domain PD5 is in Active state. Power domain PD5 is commanded to switch to Active state.

### 3.4.2 Memory Power Domain Control Register 0 (MEMPDWCTRL0)

**Figure 3-4. Memory Power Domain Control Register 0 (MEMPDWCTRL0) [offset = 10h]**

31	28	27	24	23	20	19	16
Reserved		MEMPDON0		Reserved		MEMPDON1	
R-0		R/W-n		R-0		R/W-n	
15	12	11	8	7			
Reserved		MEMPDON2		Reserved			
R-0		R/W-n		R-0			

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

The default values of the control fields are determined by the device reset configuration word stored in the TI-OTP region of flash bank 0.

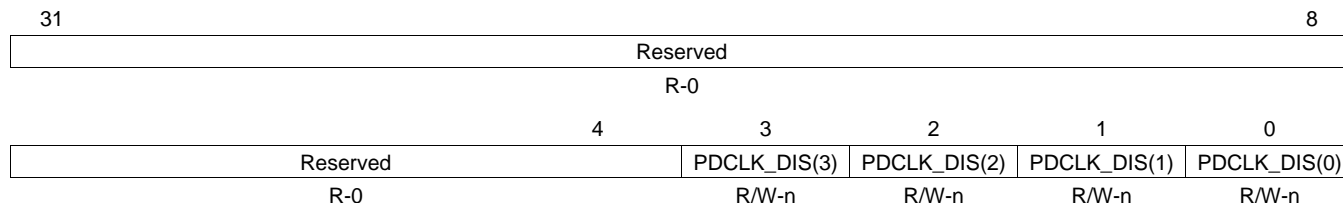
**Table 3-3. Memory Power Domain Control Register 0 (MEMPDWCTRL0) Field Descriptions**

Bit	Field	Value	Description
31-28	Reserved	0	Read returns 0. Writes have no effect.
27-24	MEMPDON0	Ah 9h Any other value	Read in User and Privileged Mode: Write in Privileged Mode only: Power domain RAM_PD1 is in OFF state. Power domain RAM_PD1 is commanded to switch to OFF state. Reserved Power domain RAM_PD1 is in Active state. Power domain RAM_PD1 is commanded to switch to Active state.
23-20	Reserved	0	Read returns 0. Writes have no effect.
19-16	MEMPDON1	Ah 9h Any other value	Read in User and Privileged Mode: Write in Privileged Mode only: Power domain RAM_PD2 is in OFF state. Power domain RAM_PD2 is commanded to switch to OFF state. Reserved Power domain RAM_PD2 is in Active state. Power domain RAM_PD2 is commanded to switch to Active state.
15-12	Reserved	0	Read returns 0. Writes have no effect.
11-8	MEMPDON2	Ah 9h Any other value	Read in User and Privileged Mode: Write in Privileged Mode only: Power domain RAM_PD3 is in OFF state. Power domain RAM_PD3 is commanded to switch to OFF state. Reserved Power domain RAM_PD3 is in Active state. Power domain RAM_PD3 is commanded to switch to Active state.
7-0	Reserved	0	Read returns 0. Writes have no effect.



### 3.4.3 Power Domain Clock Disable Register (PDCLKDISREG)

**Figure 3-5. Power Domain Clock Disable Register (PDCLKDISREG) [offset = 20h]**



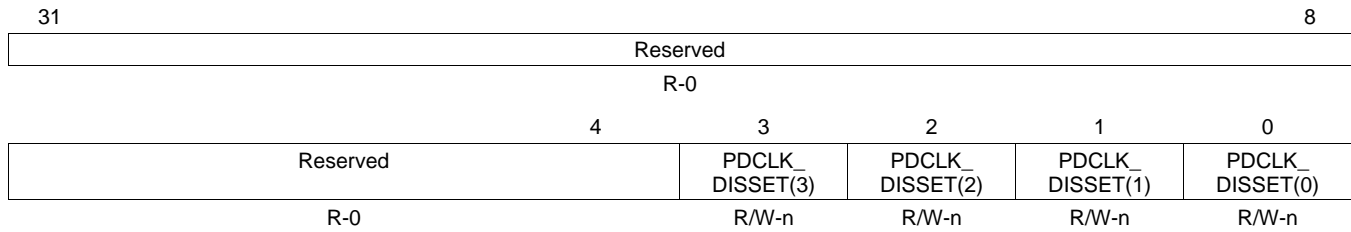
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

The default values of the control fields are determined by the device reset configuration word stored in the TI-OTP region of flash bank 0.

**Table 3-4. Power Domain Clock Disable Register (PDCLKDISREG) Field Descriptions**

Bit	Field	Value	Description
31-4	Reserved	0	Read returns 0. Writes have no effect.
3	PDCLK_DIS[3]	0	Read in User and Privileged Mode returns the current value of PDCLK_DIS[3] Write in Privileged Mode only
		1	Enable clocks to logic power domain PD5 Disable clocks to logic power domain PD5
2	PDCLK_DIS[2]	0	Read in User and Privileged Mode returns the current value of PDCLK_DIS[2] Write in Privileged Mode only
		1	Enable clocks to logic power domain PD4 Disable clocks to logic power domain PD4
1	PDCLK_DIS[1]	0	Read in User and Privileged Mode returns the current value of PDCLK_DIS[1] Write in Privileged Mode only
		1	Enable clocks to logic power domain PD3 Disable clocks to logic power domain PD3
0	PDCLK_DIS[0]	0	Read in User and Privileged Mode returns the current value of PDCLK_DIS[0] Write in Privileged Mode only
		1	Enable clocks to logic power domain PD2 Disable clocks to logic power domain PD2

### 3.4.4 Power Domain Clock Disable Set Register (PDCLKDISSETREG)

**Figure 3-6. Power Domain Clock Disable Set Register (PDCLKDISSETREG) [offset = 24h]**


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

The default values of the control fields are determined by the device reset configuration word stored in the TI-OTP region of flash bank 0.

**Table 3-5. Power Domain Clock Disable Set Register (PDCLKDISSETREG) Field Descriptions**

Bit	Field	Value	Description
31-4	Reserved	0	Read returns 0. Writes have no effect.
3	PDCLK_DISSET [3]	0 1	Read in User and Privileged Mode returns the current value of PDCLK_DISSET[3] Write in Privileged Mode only 0 No effect to state of clocks to power domain PD5 1 Disable clocks to logic power domain PD5
2	PDCLK_DISSET [2]	0 1	Read in User and Privileged Mode returns the current value of PDCLK_DISSET[2] Write in Privileged Mode only 0 No effect to state of clocks to power domain PD4 1 Disable clocks to logic power domain PD4
1	PDCLK_DISSET [1]	0 1	Read in User and Privileged Mode returns the current value of PDCLK_DISSET[1] Write in Privileged Mode only 0 No effect to state of clocks to power domain PD3 1 Disable clocks to logic power domain PD3
0	PDCLK_DISSET [0]	0 1	Read in User and Privileged Mode returns the current value of PDCLK_DISSET[0] Write in Privileged Mode only 0 No effect to state of clocks to power domain PD2 1 Disable clocks to logic power domain PD2

### 3.4.5 Power Domain Clock Disable Clear Register (PDCLKDISCLRREG)

**Figure 3-7. Power Domain Clock Disable Clear Register (PDCLKDISCLRREG) [offset = 28h]**

31	Reserved				8
R-0					
4		3	2	1	0
Reserved		PDCLK_ DISCLR(3)	PDCLK_ DISCLR(2)	PDCLK_ DISCLR(1)	PDCLK_ DISCLR(0)
R-0		R/W-n	R/W-n	R/W-n	R/W-n

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

The default values of the control fields are determined by the device reset configuration word stored in the TI-OTP region of flash bank 0.

**Table 3-6. Power Domain Clock Disable Clear Register (PDCLKDISCLRREG) Field Descriptions**

Bit	Field	Value	Description
31-4	Reserved	0	Read returns 0. Writes have no effect.
3	PDCLK_DISCLR[3]	0	Read in User and Privileged Mode returns the current value of PDCLK_DIS[3] Write in Privileged Mode only No effect to state of clocks to power domain PD5
		1	Enable clocks to logic power domain PD5
2	PDCLK_DISCLR[2]	0	Read in User and Privileged Mode returns the current value of PDCLK_DIS[2] Write in Privileged Mode only No effect to state of clocks to power domain PD4
		1	Enable clocks to logic power domain PD4
1	PDCLK_DISCLR[1]	0	Read in User and Privileged Mode returns the current value of PDCLK_DIS[1] Write in Privileged Mode only No effect to state of clocks to power domain PD3
		1	Enable clocks to logic power domain PD3
0	PDCLK_DISCLR[0]	0	Read in User and Privileged Mode returns the current value of PDCLK_DIS[0] Write in Privileged Mode only No effect to state of clocks to power domain PD2
		1	Enable clocks to logic power domain PD2

### 3.4.6 Logic Power Domain PD2 Power Status Register (LOGICPDPWRSTAT0)

**Figure 3-8. Logic Power Domain PD2 Power Status Register (LOGICPDPWRSTAT0) [offset = 40h]**

31	25	24	23	17	16
Reserved		LOGIC IN TRANS0	Reserved		MEM IN TRANS0
R-0		R-n	R-0		R-n
15	9	8	7	2	1 0
Reserved		DOMAIN ON0	Reserved		LOGICPDPWR STAT0
R-0		R-n	R-0		R-n

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

This is a read-only register. All writes are ignored.

The default values of the control fields are determined by the device reset configuration word stored in the TI-OTP region of flash bank 0.

**Table 3-7. Logic Power Domain PD2 Power Status Register (LOGICPDPWRSTAT0) Field Descriptions**

Bit	Field	Value	Description
31-25	Reserved	0	Read returns 0. Writes have no effect.
24	LOGIC IN TRANS0	0	Logic in transition status for power domain PD2 Read in User and Privileged Mode Logic in power domain PD2 is in the steady Active or OFF state
		1	Logic in power domain PD2 is in the process of power-down/up
13-17	Reserved	0	Read returns 0. Writes have no effect.
16	MEM IN TRANS0	0	Memory in transition status for power domain PD2 Read in User and Privileged Mode Memory in power domain PD2 is in the steady Active or OFF state
		1	Memory in power domain PD2 is in the process of power-down/up
15-9	Reserved	0	Read returns 0. Writes have no effect.
8	DOMAIN ON0	0	Current state of power domain PD2. The default value of this field is controlled by the device reset configuration word in the TI-OTP region of flash bank 0. Read in User and Privileged Mode Power domain PD2 is in the OFF state
		1	Power domain PD2 is in the Active state
7-2	Reserved	0	Read returns 0. Writes have no effect.
1-0	LOGICPDPWR STAT0	0	Logic power domain PD2 power state. The default value of this field is controlled by the device reset configuration word in the TI-OTP region of flash bank 0. Read in User and Privileged Mode Logic power domain PD2 is switched OFF
		1h	Logic power domain PD2 is in Idle state
		2h	Reserved
		3h	Logic power domain PD2 is in Active state

### 3.4.7 Logic Power Domain PD3 Power Status Register (LOGICPDPWRSTAT1)

**Figure 3-9. Logic Power Domain PD3 Power Status Register (LOGICPDPWRSTAT1) [offset = 44h]**

31	25	24	23	17	16
Reserved		LOGIC IN TRANS1	Reserved		MEM IN TRANS1
R-0		R-n	R-0		R-n
15	9	8	7	2	1 0
Reserved		DOMAIN ON1	Reserved		LOGICPDPWR STAT1
R-0		R-n	R-0		R-n

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

This is a read-only register. All writes are ignored.

The default values of the control fields are determined by the device reset configuration word stored in the TI-OTP region of flash bank 0.

**Table 3-8. Logic Power Domain PD3 Power Status Register (LOGICPDPWRSTAT1) Field Descriptions**

Bit	Field	Value	Description
31-25	Reserved	0	Read returns 0. Writes have no effect.
24	LOGIC IN TRANS1	0	Logic in transition status for power domain PD3 Read in User and Privileged Mode Logic in power domain PD3 is in the steady Active or OFF state
		1	Logic in power domain PD3 is in the process of power-down/up
13-17	Reserved	0	Read returns 0. Writes have no effect.
16	MEM IN TRANS1	0	Memory in transition status for power domain PD3 Read in User and Privileged Mode Memory in power domain PD3 is in the steady Active or OFF state
		1	Memory in power domain PD3 is in the process of power-down/up
15-9	Reserved	0	Read returns 0. Writes have no effect.
8	DOMAIN ON1	0	Current state of power domain PD3. The default value of this field is controlled by the device reset configuration word in the TI-OTP region of flash bank 0. Read in User and Privileged Mode Power domain PD3 is in the OFF state
		1	Power domain PD3 is in the Active state
7-2	Reserved	0	Read returns 0. Writes have no effect.
1-0	LOGICPDPWR STAT1	0	Logic power domain PD3 power state. The default value of this field is controlled by the device reset configuration word in the TI-OTP region of flash bank 0. Read in User and Privileged Mode Logic power domain PD3 is switched OFF
		1h	Logic power domain PD3 is in Idle state
		2h	Reserved
		3h	Logic power domain PD3 is in Active state

### 3.4.8 Logic Power Domain PD4 Power Status Register (LOGICPDPWRSTAT2)

**Figure 3-10. Logic Power Domain PD4 Power Status Register (LOGICPDPWRSTAT2) [offset = 48h]**

31	25	24	23	17	16
Reserved		LOGIC IN TRANS2	Reserved		MEM IN TRANS2
R-0		R-n	R-0		R-n
15	9	8	7	2	1 0
Reserved		DOMAIN ON2	Reserved		LOGICPDPWR STAT2
R-0		R-n	R-0		R-n

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

This is a read-only register. All writes are ignored.

The default values of the control fields are determined by the device reset configuration word stored in the TI-OTP region of flash bank 0.

**Table 3-9. Logic Power Domain PD4 Power Status Register (LOGICPDPWRSTAT2) Field Descriptions**

Bit	Field	Value	Description
31-25	Reserved	0	Read returns 0. Writes have no effect.
24	LOGIC IN TRANS2	0	Logic in transition status for power domain PD4 Read in User and Privileged Mode Logic in power domain PD4 is in the steady Active or OFF state
		1	Logic in power domain PD4 is in the process of power-down/up
13-17	Reserved	0	Read returns 0. Writes have no effect.
16	MEM IN TRANS2	0	Memory in transition status for power domain PD4 Read in User and Privileged Mode Memory in power domain PD4 is in the steady Active or OFF state
		1	Memory in power domain PD4 is in the process of power-down/up
15-9	Reserved	0	Read returns 0. Writes have no effect.
8	DOMAIN ON2	0	Current state of power domain PD4. The default value of this field is controlled by the device reset configuration word in the TI-OTP region of flash bank 0. Read in User and Privileged Mode Power domain PD4 is in the OFF state
		1	Power domain PD4 is in the Active state
7-2	Reserved	0	Read returns 0. Writes have no effect.
1-0	LOGICPDPWR STAT2	0	Logic power domain PD4 power state. The default value of this field is controlled by the device reset configuration word in the TI-OTP region of flash bank 0. Read in User and Privileged Mode Logic power domain PD4 is switched OFF
		1h	Logic power domain PD4 is in Idle state
		2h	Reserved
		3h	Logic power domain PD4 is in Active state

### 3.4.9 Logic Power Domain PD5 Power Status Register (LOGICPDPWRSTAT3)

**Figure 3-11. Logic Power Domain PD5 Power Status Register (LOGICPDPWRSTAT3) [offset = 4Ch]**

31	25	24	23	17	16
Reserved		LOGIC IN TRANS3	Reserved		MEM IN TRANS3
R-0		R-n	R-0		R-n
15	9	8	7	2	1 0
Reserved		DOMAIN ON3	Reserved		LOGICPDPWR STAT3
R-0		R-n	R-0		R-n

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

This is a read-only register. All writes are ignored.

The default values of the control fields are determined by the device reset configuration word stored in the TI-OTP region of flash bank 0.

**Table 3-10. Logic Power Domain PD5 Power Status Register (LOGICPDPWRSTAT3) Field Descriptions**

Bit	Field	Value	Description
31-25	Reserved	0	Read returns 0. Writes have no effect.
24	LOGIC IN TRANS3	0	Logic in transition status for power domain PD5 Read in User and Privileged Mode Logic in power domain PD5 is in the steady Active or OFF state
		1	Logic in power domain PD5 is in the process of power-down/up
13-17	Reserved	0	Read returns 0. Writes have no effect.
16	MEM IN TRANS3	0	Memory in transition status for power domain PD5 Read in User and Privileged Mode Memory in power domain PD5 is in the steady Active or OFF state
		1	Memory in power domain PD5 is in the process of power-down/up
15-9	Reserved	0	Read returns 0. Writes have no effect.
8	DOMAIN ON3	0	Current state of power domain PD5. The default value of this field is controlled by the device reset configuration word in the TI-OTP region of flash bank 0. Read in User and Privileged Mode Power domain PD5 is in the OFF state
		1	Power domain PD5 is in the Active state
7-2	Reserved	0	Read returns 0. Writes have no effect.
1-0	LOGICPDPWR STAT3	0	Logic power domain PD5 power state. The default value of this field is controlled by the device reset configuration word in the TI-OTP region of flash bank 0. Read in User and Privileged Mode Logic power domain PD5 is switched OFF
		1h	Logic power domain PD5 is in Idle state
		2h	Reserved
		3h	Logic power domain PD5 is in Active state

### 3.4.10 Memory Power Domain RAM\_PD1 Power Status Register (MEMPDWRSTAT0)

**Figure 3-12. Memory Power Domain RAM\_PD1 Power Status Register (MEMPDWRSTAT0) [offset = 80h]**

31	25	24	23	17	16
Reserved	LOGIC IN TRANS0		Reserved	MEM IN TRANS0	
R-0	R-n		R-0	R-n	
15	9	8	7	2	1 0
Reserved	DOMAIN ON0		Reserved	MEMPDWR STAT0	
R-0	R-n		R-0	R-n	

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

This is a read-only register. All writes are ignored.

The default values of the control fields are determined by the device reset configuration word stored in the TI-OTP region of flash bank 0.

**Table 3-11. Memory Power Domain RAM\_PD1 Power Status Register (MEMPDWRSTAT0) Field Descriptions**

Bit	Field	Value	Description
31-25	Reserved	0	Read returns 0. Writes have no effect.
24	LOGIC IN TRANS0	0 1	Logic in transition status for power domain RAM_PD1. This power domain only contains SRAM macros. However, an SRAM macro also has some digital logic controlled by the PSCON. Therefore, a memory power domain also contains a logic status indicator.  Read in User and Privileged Mode Logic in power domain RAM_PD1 is in the steady Active or OFF state Logic in power domain RAM_PD1 is in the process of power-down/up
13-17	Reserved	0	Read returns 0. Writes have no effect.
16	MEM IN TRANS0	0 1	Memory in transition status for power domain RAM_PD1 Read in User and Privileged Mode Memory in power domain RAM_PD1 is in the steady Active or OFF state Memory in power domain RAM_PD1 is in the process of power-down/up
15-9	Reserved	0	Read returns 0. Writes have no effect.
8	DOMAIN ON0	0 1	Current state of power domain RAM_PD1. The default value of this field is controlled by the device reset configuration word in the TI-OTP region of flash bank 0.  Read in User and Privileged Mode Power domain RAM_PD1 is in the OFF state Power domain RAM_PD1 is in the Active state
7-2	Reserved	0	Read returns 0. Writes have no effect.
1-0	MEMPDWR STAT0	0 1h 2h 3h	Memory power domain RAM_PD1 power state. The default value of this field is controlled by the device reset configuration word in the TI-OTP region of flash bank 0. Read in User and Privileged Mode Logic power domain RAM_PD1 is switched OFF Logic power domain RAM_PD1 is in Idle state Reserved Logic power domain RAM_PD1 is in Active state



### 3.4.11 Memory Power Domain RAM\_PD2 Power Status Register (MEMPDWRSTAT1)

**Figure 3-13. Memory Power Domain RAM\_PD2 Power Status Register (MEMPDWRSTAT1) [offset = 84h]**

31	25	24	23	17	16
Reserved		LOGIC IN TRANS1	Reserved		MEM IN TRANS1
R-0		R-n	R-0		R-n
15	9	8	7	2	1 0
Reserved		DOMAIN ON1	Reserved		MEMPDWR STAT1
R-0		R-n	R-0		R-n

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

This is a read-only register. All writes are ignored.

The default values of the control fields are determined by the device reset configuration word stored in the TI-OTP region of flash bank 0.

**Table 3-12. Memory Power Domain RAM\_PD2 Power Status Register (MEMPDWRSTAT1) Field Descriptions**

Bit	Field	Value	Description
31-25	Reserved	0	Read returns 0. Writes have no effect.
24	LOGIC IN TRANS1	0 1	Logic in transition status for power domain RAM_PD2. This power domain only contains SRAM macros. However, an SRAM macro also has some digital logic controlled by the PSCON. Therefore, a memory power domain also contains a logic status indicator.  Read in User and Privileged Mode Logic in power domain RAM_PD2 is in the steady Active or OFF state Logic in power domain RAM_PD2 is in the process of power-down/up
13-17	Reserved	0	Read returns 0. Writes have no effect.
16	MEM IN TRANS1	0 1	Memory in transition status for power domain RAM_PD2 Read in User and Privileged Mode Memory in power domain RAM_PD2 is in the steady Active or OFF state Memory in power domain RAM_PD2 is in the process of power-down/up
15-9	Reserved	0	Read returns 0. Writes have no effect.
8	DOMAIN ON1	0 1	Current state of power domain RAM_PD2. The default value of this field is controlled by the device reset configuration word in the TI-OTP region of flash bank 0.  Read in User and Privileged Mode Power domain RAM_PD2 is in the OFF state Power domain RAM_PD2 is in the Active state
7-2	Reserved	0	Read returns 0. Writes have no effect.
1-0	MEMPDWR STAT1	0 1h 2h 3h	Memory power domain RAM_PD2 power state. The default value of this field is controlled by the device reset configuration word in the TI-OTP region of flash bank 0. Read in User and Privileged Mode Logic power domain RAM_PD2 is switched OFF Logic power domain RAM_PD2 is in Idle state Reserved Logic power domain RAM_PD2 is in Active state

### 3.4.12 Memory Power Domain RAM\_PD3 Power Status Register (MEMPDWRSTAT2)

**Figure 3-14. Memory Power Domain RAM\_PD3 Power Status Register (MEMPDWRSTAT2) [offset = 88h]**

31	25	24	23	17	16
Reserved		LOGIC IN TRANS2	Reserved		MEM IN TRANS2
R-0		R-n	R-0		R-n
15	9	8	7	2	1 0
Reserved		DOMAIN ON2	Reserved		MEMPDWR STAT2
R-0		R-n	R-0		R-n

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

This is a read-only register. All writes are ignored.

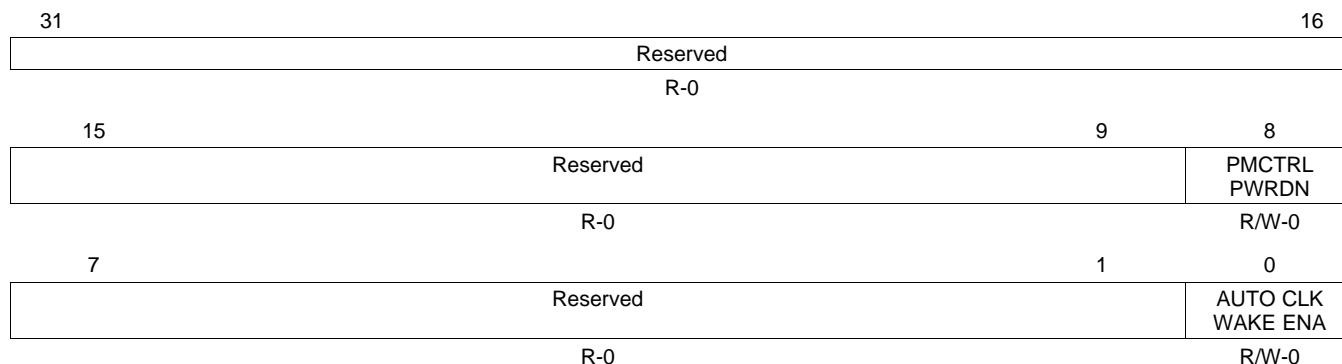
The default values of the control fields are determined by the device reset configuration word stored in the TI-OTP region of flash bank 0.

**Table 3-13. Memory Power Domain RAM\_PD3 Power Status Register (MEMPDWRSTAT2) Field Descriptions**

Bit	Field	Value	Description
31-25	Reserved	0	Read returns 0. Writes have no effect.
24	LOGIC IN TRANS2	0 1	Logic in transition status for power domain RAM_PD3. This power domain only contains SRAM macros. However, an SRAM macro also has some digital logic controlled by the PSCON. Therefore, a memory power domain also contains a logic status indicator.  Read in User and Privileged Mode Logic in power domain RAM_PD3 is in the steady Active or OFF state Logic in power domain RAM_PD3 is in the process of power-down/up
13-17	Reserved	0	Read returns 0. Writes have no effect.
16	MEM IN TRANS2	0 1	Memory in transition status for power domain RAM_PD3 Read in User and Privileged Mode Memory in power domain RAM_PD3 is in the steady Active or OFF state Memory in power domain RAM_PD3 is in the process of power-down/up
15-9	Reserved	0	Read returns 0. Writes have no effect.
8	DOMAIN ON2	0 1	Current state of power domain RAM_PD3. The default value of this field is controlled by the device reset configuration word in the TI-OTP region of flash bank 0.  Read in User and Privileged Mode Power domain RAM_PD3 is in the OFF state Power domain RAM_PD3 is in the Active state
7-2	Reserved	0	Read returns 0. Writes have no effect.
1-0	MEMPDWR STAT2	0 1h 2h 3h	Memory power domain RAM_PD3 power state. The default value of this field is controlled by the device reset configuration word in the TI-OTP region of flash bank 0.  Read in User and Privileged Mode Logic power domain RAM_PD3 is switched OFF Logic power domain RAM_PD3 is in Idle state Reserved Logic power domain RAM_PD3 is in Active state

### 3.4.13 Global Control Register 1 (GLOBALCTRL1)

Figure 3-15. Global Control Register 1 (GLOBALCTRL1) [offset = A0h]

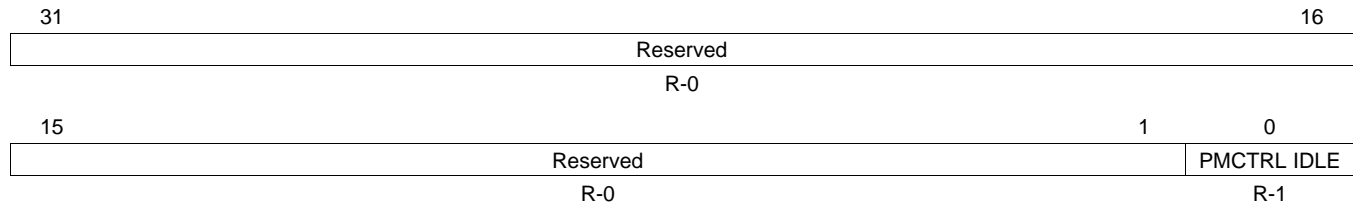


LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 3-14. Global Control Register 1 (GLOBALCTRL1) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Read returns 0. Writes have no effect.
8	PMCTRL PWRDN	0	PMC/PSCON Power Down Read in User and Privileged Mode returns current value of PMCTRL PWRDN Write in Privileged mode only Enable the clock to pmctrl_wakeup block.
		1	Disable the clock to pmctrl_wakeup block, which contains PMC and all PSCONs.
7-1	Reserved	0	Read returns 0. Writes have no effect.
0	AUTO CLK WAKE ENA	0	Automatic Clock Enable on Wake Up Read in User and Privileged Mode returns current value of AUTO CLK WAKE ENA Write in Privileged mode only Disable automatic clock wake up. The application must enable clocks by clearing the correct bit in the PDCLK_DIS register.
		1	Enable automatic clock wake up when a power domain transitions to Active state.

### 3.4.14 Global Status Register (GLOBALSTAT)

**Figure 3-16. Global Status Register (GLOBALSTAT) [offset = A8h]**


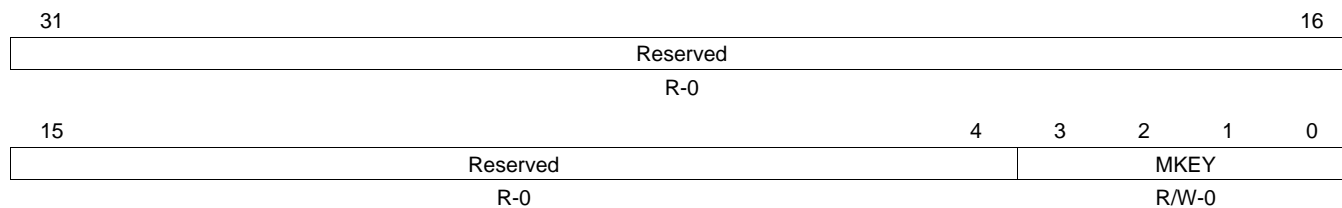
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 3-15. Global Status Register (GLOBALSTAT) Field Descriptions**

Bit	Field	Value	Description
31-1	Reserved	0	Read returns 0. Writes have no effect.
0	PMCTRL IDLE	0	PMC and PSCONs for all power domains are in the process of generating power state transition control sequence for logic and/or SRAM.
		1	PMC and PSCONs for all power domains have completed generating power state transition control sequence triggered by PMC input control signals.

### 3.4.15 PSCON Diagnostic Compare Key Register (PRCKEYREG)

Figure 3-17. PSCON Diagnostic Compare Key Register (PRCKEYREG) [offset = ACh]



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

Table 3-16. PSCON Diagnostic Compare Key Register (PRCKEYREG) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Read returns 0. Writes have no effect.
3-0	MKEY		Diagnostic PSCON Mode Key. The mode key is applied to all individual PSCON compare units. Read in User and Privileged mode returns the current value of MKEY Write in Privileged mode only
		0	Lock Step mode
		6h	Self Test mode
		9h	Error Forcing mode
		Fh	Self Test Error Forcing Mode
		All others	Lock Step mode

### 3.4.16 LogicPD PSCON Diagnostic Compare Status Register 1 (LPDDCSTAT1)

**Figure 3-18. LogicPD PSCON Diagnostic Compare Status Register 1 (LPDDCSTAT1) [offset = B0h]**

31	Reserved				24
R-0					
23	20	19	18	17	16
Reserved		LCMPE [3]	LCMPE [2]	LCMPE [1]	LCMPE [0]
R-0		RC-0	RC-0	RC-0	RC-0
15	Reserved				8
R-0					
7	4	3	2	1	0
Reserved		LSTC[3]	LSTC[2]	LSTC[1]	LSTC[0]
R-0		R-0	R-0	R-0	R-0

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

**Table 3-17. LogicPD PSCON Diagnostic Compare Status Register 1 (LPDDCSTAT1) Field Descriptions**

Bit	Field	Value	Description
31-20	Reserved	0	Read returns 0. Writes have no effect.
19-16	LCMPE[3-0]	0 1	Logic Power Domain Compare Error Each of these bits corresponds to a logic power domain: Bit 3 for PD5, Bit 2 for PD4, Bit 1 for PD3, Bit 0 for PD2.  Read in User and Privileged Mode Write in Privileged mode only  0 PSCON signals are identical. Writing 0 has no effect. 1 Read: PSCON signal compare mismatch identified. Write: Clears the corresponding LCMPE bit, if set.
15-4	Reserved	0	Read returns 0. Writes have no effect.
3-0	LSTC[3-0]	0 1	Logic Power Domain Self Test Complete Each of these bits corresponds to a logic power domain: Bit 3 for PD5, Bit 2 for PD4, Bit 1 for PD3, Bit 0 for PD2.  Read in User and Privileged Mode Writes have no effect  0 Self test is ongoing if self-test mode is entered. 1 Self test is complete.

### 3.4.17 LogicPD PSCON Diagnostic Compare Status Register 2 (LPDDCSTAT2)

**Figure 3-19. LogicPD PSCON Diagnostic Compare Status Register 2 (LPDDCSTAT2) [offset = B4h]**

31	Reserved				24
R-0					
23	20	19	18	17	16
Reserved		LSTET[3]	LSTET[2]	LSTET[1]	LSTET[0]
R-0		R-0	R-0	R-0	R-0
15	Reserved				8
R-0					
7	4	3	2	1	0
Reserved		LSTE[3]	LSTE[2]	LSTE[1]	LSTE[0]
R-0		R-0	R-0	R-0	R-0

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

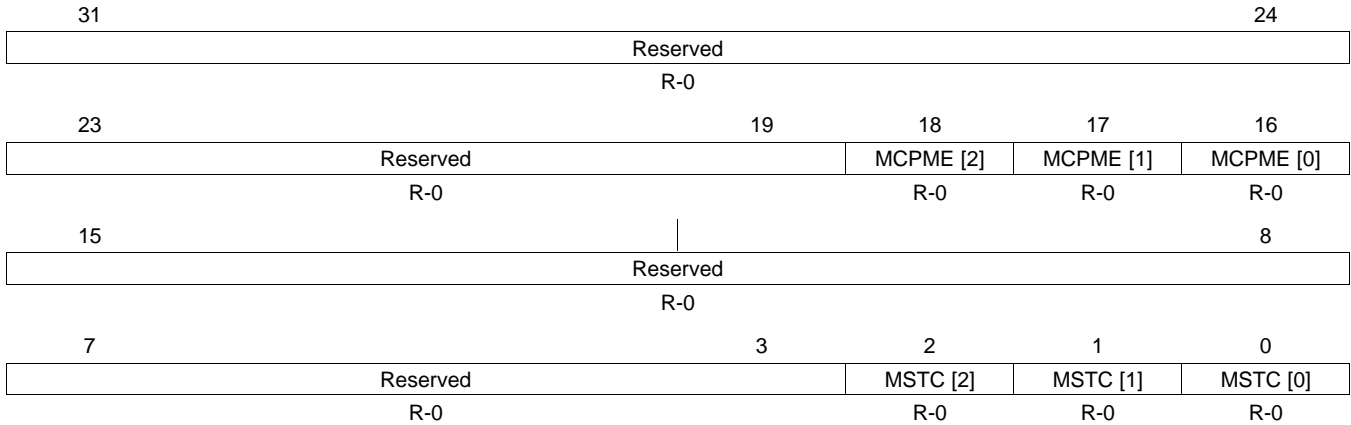
**Table 3-18. LogicPD PSCON Diagnostic Compare Status Register 2 (LPDDCSTAT2) Field Descriptions**

Bit	Field	Value	Description
31-20	Reserved	0	Read returns 0. Writes have no effect.
19-16	LSTET[3-0]	0	Logic Power Domain Self Test Error Type Each of these bits corresponds to a logic power domain: Bit 3 for PD5, Bit 2 for PD4, Bit 1 for PD3, Bit 0 for PD2.  Read in User and Privileged Mode Writes have no effect
		1	Self test failed during compare mismatch test.
15-4	Reserved	0	Read returns 0. Writes have no effect.
3-0	LSTE[3-0]	0	Logic Power Domain Self Test Error Each of these bits corresponds to a logic power domain: Bit 3 for PD5, Bit 2 for PD4, Bit 1 for PD3, Bit 0 for PD2.  Read in User and Privileged Mode Writes have no effect
		1	Self test failed.

### 3.4.18 Memory PD PSCON Diagnostic Compare Status Register 1 (MPDDCSTAT1)

This register shows the interrupt status (before enabling) and allows setting of the interrupt status.

**Figure 3-20. Memory PD PSCON Diagnostic Compare Status Register 1 (MPDDCSTAT1) [offset = B8h]**



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

**Table 3-19. Memory PD PSCON Diagnostic Compare Status Register 1 (MPDDCSTAT1) Field Descriptions**

Bit	Field	Value	Description
31-19	Reserved	0	Read returns 0. Writes have no effect.
18-16	MCMPE[2-0]	0 1	Memory Power Domain Compare Error Each of these bits corresponds to a memory power domain: Bit 2 for RAM_PD3, Bit 1 for RAM_PD2, Bit 0 for RAM_PD1.  Read in User and Privileged Mode Write in Privileged mode only  0 PSCON signals are identical. Writing 0 has no effect. 1 Read: PSCON signal compare mismatch identified. Write: Clears the corresponding MCMPE bit, if set.
15-3	Reserved	0	Read returns 0. Writes have no effect.
2-0	MSTC[2-0]	0 1	Memory Power Domain Self Test Complete Each of these bits corresponds to a memory power domain: Bit 2 for RAM_PD3, Bit 1 for RAM_PD2, Bit 0 for RAM_PD1.  Read in User and Privileged Mode Writes have no effect  0 Self test is ongoing if self-test mode is entered. 1 Self test is complete.



### 3.4.19 Memory PD PSCON Diagnostic Compare Status Register 2 (MPDDCSTAT2)

Figure 3-21. Memory PD PSCON Diagnostic Compare Status Register 2 (MPDDCSTAT2) [offset = BCh]

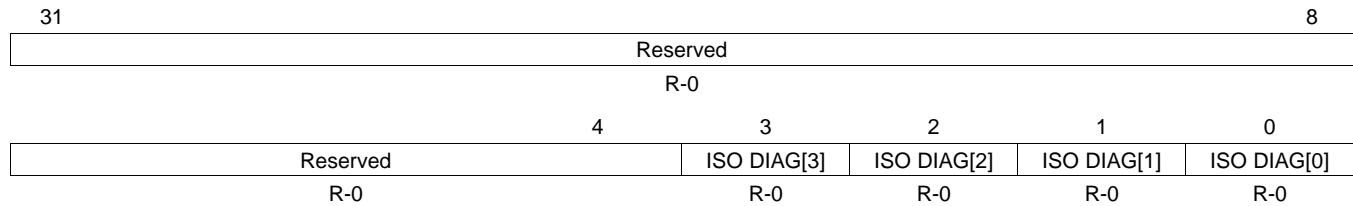
31	Reserved				24
R-0					
23	19	18	17	16	
Reserved		MSTET[2]	MSTET[1]	MSTET[0]	
R-0		R-0	R-0	R-0	
15	Reserved			8	
R-0					
7	3	2	1	0	
Reserved		MSTE [2]	MSTE [1]	MSTE [0]	
R-0		R-0	R-0	R-0	

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

Table 3-20. Memory PD PSCON Diagnostic Compare Status Register 2 (MPDDCSTAT2) Field Descriptions

Bit	Field	Value	Description
31-19	Reserved	0	Read returns 0. Writes have no effect.
18-16	MSTET[2-0]	0	Memory Power Domain Self Test Error Type Each of these bits corresponds to a memory power domain: Bit 2 for RAM_PD3, Bit 1 for RAM_PD2, Bit 0 for RAM_PD1.  Read in User and Privileged Mode Writes have no effect
		1	Self test failed during compare mismatch test.
15-3	Reserved	0	Read returns 0. Writes have no effect.
2-0	MSTE[2-0]	0	Memory Power Domain Self Test Error Each of these bits corresponds to a memory power domain: Bit 2 for RAM_PD3, Bit 1 for RAM_PD2, Bit 0 for RAM_PD1.  Read in User and Privileged Mode Writes have no effect
		1	Self test failed.

### 3.4.20 Isolation Diagnostic Status Register (ISODIAGSTAT)

**Figure 3-22. Isolation Diagnostic Status Register (ISODIAGSTAT) [offset = C0h]**


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

**Table 3-21. Isolation Diagnostic Status Register (ISODIAGSTAT) Field Descriptions**

Bit	Field	Value	Description
31-4	Reserved	0	Read returns 0. Writes have no effect.
3-0	ISO DIAG[3-0]		Isolation Diagnostic Each of these bits corresponds to a logic power domain: Bit 3 for PD5, Bit 2 for PD4, Bit 1 for PD3, Bit 0 for PD2.  Read in User and Privileged Mode Writes have no effect
		0	Isolation is enabled for corresponding power domain
		1	Isolation is disabled for corresponding power domain



## ***I/O Multiplexing and Control Module (IOMM)***

---



---

This chapter describes the I/O Multiplexing and Control Module (IOMM).

Topic	Page
<b>4.1 Overview .....</b>	<b>250</b>
<b>4.2 Main Features of I/O Multiplexing Module (IOMM) .....</b>	<b>250</b>
<b>4.3 Control of Multiplexed Functions .....</b>	<b>251</b>
<b>4.4 Safety Features .....</b>	<b>254</b>
<b>4.5 IOMM Registers .....</b>	<b>255</b>
<b>4.6 Signal Multiplexing and Control on Microcontrollers .....</b>	<b>268</b>

## 4.1 Overview

This chapter describes the overall features of the module which controls the I/O multiplexing on the device. The mapping of control registers to multiplexing options is specified in [Section 4.6](#).

## 4.2 Main Features of I/O Multiplexing Module (IOMM)

The IOMM contains memory-mapped registers (MMR) that control device-specific multiplexed functions. The safety and diagnostic features of the IOMM are:

- Kicker mechanism to protect the MMRs from accidental writes
- Master-id checker to only allow the CPU to write to the MMRs
- Error indication for access violations

### 4.3 Control of Multiplexed Functions

Several functions are multiplexed on the microcontrollers. The following sections describe the multiplexing scheme and its implementation.

#### 4.3.1 Control of Multiplexed Outputs

The signal multiplexing controlled by each memory-mapped control register (PINMMRn) is described in [Table 4-15](#). Each byte in the PINMMRs control the functionality output on a single terminal. Consider the following example for the PINMMR10 control register.

**Figure 4-1. PINMMR10 Control Register, Address = FFFF EB38h**

31	26	25	24	23	18	17	16	
Reserved		EMIF DATA[ 2]	ETM DATA [18]	Reserved		RTP DATA[ 15]	EMIF nCS[0]	
R-0		R/W-0	R/W-1	R-0		R/W-0	R/W-1	
15	10	9	8	7	3	2	1	0
Reserved		EMIF DATA[ 3]	ETM DATA [19]	Reserved		RMII_ RX_E R	MII_R X_ER	AD1E VT
R-0		R/W-0	R/W-1	R-0		R/W-0	R/W-0	R/W-1

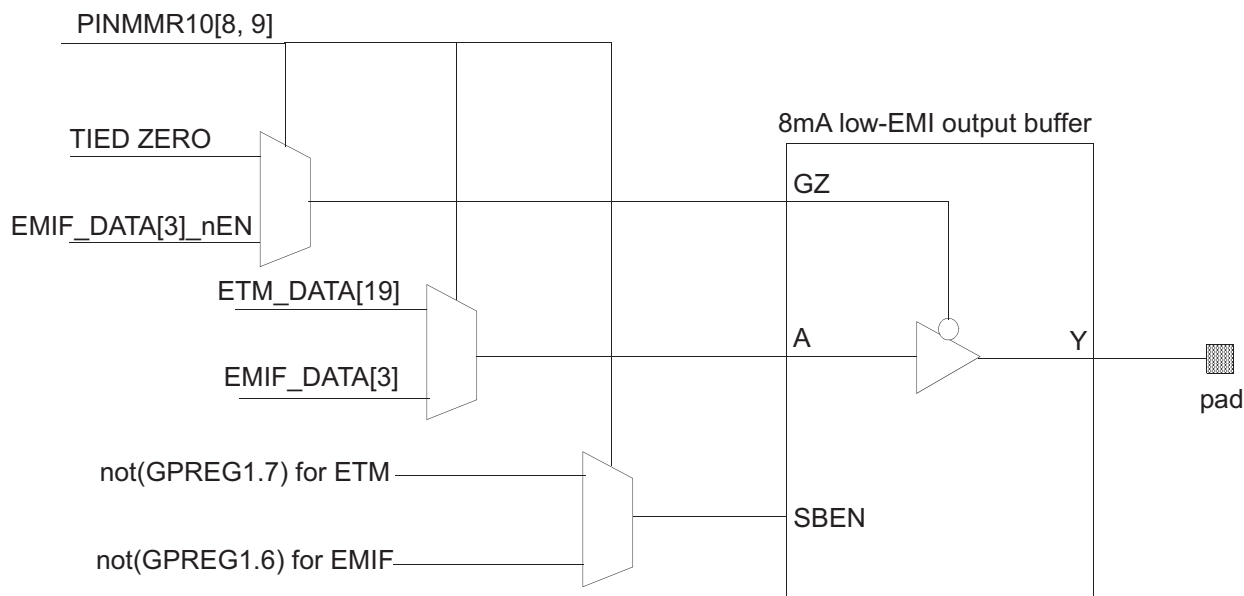
LEGEND: R/W = Read/Write; R = Read only; C = Clear; -n = value after reset

- Consider the multiplexing controlled by PINMMR10[15–8]. These bits control the multiplexing between the ETMDATA[19] and EMIF\_DATA[3] on the ball N15 of the 337BGA package for this device. The default function on the N15 ball is ETMDATA[19]. This is indicated by bit 8 of the PINMMR10 register being set.
- If the application wants to use N15 as an EMIF\_DATA[3] signal, then bit 8 of PINMMR10 must be cleared and bit 9 must be set.
- Each feature of the output function is determined by the function selected to be output on a terminal.

For example, the ball N15 on the 337BGA package is driven by an output buffer with an 8mA drive strength and which supports the adaptive impedance control mode for reduced electromagnetic emissions. This output buffer has the following signals: A (signal to be output), GZ (output enable), SBEN (Standard Buffer Enable) and LPM (Low Power Mode). Each of these signals is an output of a multiplexor which allows the selected function to control all available features of the output buffer.

- The PINMMR control registers are used to implement a one-hot encoding scheme for selecting the multiplexed function.
  - For example, for the N15 ball on the 337BGA package for this device at least one out of bit 8 or bit 9 must be set.
  - If the application clears both bits 8 and 9, then the default function will be selected for output on N15.
  - If the application sets both bits 8 and 9, then the default function will be selected for output on N15.
  - If the application sets one or more reserved bit(s) within the byte 15–8, then the default function will be selected for output on N15.

[Figure 4-2](#) shows the multiplexing between the output functions for the N15 ball. This terminal uses an 8mA low-EMI output buffer.

**Figure 4-2. Output Multiplexing Example**

Notes on [Figure 4-2](#):

- ETM\_DATA[19] is an output-only signal. Therefore the GZ is tied to zero when the ETM functionality is selected to be output on this pad.
- The low-EMI mode controls for the EMIF and ETM signals are independent. These controls are multiplexed and chosen as per the function being output on this pad.

### 4.3.2 Control of Multiplexed Inputs

The microcontrollers are available in two package options – 337-ball grid array (BGA) and 144-pin Quad Flat Pack (QFP). Input multiplexing is not required for the 337BGA package. For the 144QFP package, some signals have a multiplexor implemented on their input side.

**Table 4-1. Input Multiplexing on 144QFP Parts**

Signal	144QFP Dedicated Input Pin #	144QFP Multiplexed Input Pin #	A, Output Control for Muxed Pin	B, Input Control for Muxed Pin
SPI4SIMO	–	30	PINMMR5[9]	PINMMR23[16]
SPI4SOMI	–	31	PINMMR5[17]	PINMMR23[24]
SPI4CLK	–	25	PINMMR5[1]	PINMMR23[8]
SPI4NENA	–	23	PINMMR4[17]	PINMMR24[0]
SPI4NCS[0]	–	24	PINMMR4[25]	PINMMR24[8]
N2HET1[17]	–	130	PINMMR20[17]	PINMMR24[16]
N2HET1[19]	–	40	PINMMR8[9]	PINMMR24[24]
N2HET1[23]	–	96	PINMMR12[17]	PINMMR25[8]
N2HET1[25]	–	37	PINMMR7[9]	PINMMR25[16]
N2HET1[27]	–	4	PINMMR0[26]	PINMMR25[24]
N2HET1[29]	–	3	PINMMR0[18]	PINMMR26[0]
N2HET1[31]	–	54	PINMMR9[10]	PINMMR26[8]
GIOB[2]	142	55	PINMMR29[16]	PINMMR29[16]

For signals with a “–” in the column for dedicated input pin #, these signals do have a dedicated pad on the die. The default input for these signals comes from the dedicated pad. To be able to drive in the input from the multiplexed pins on the 144QFP package, the IOMM registers need to be configured.

Table 4-1 shows two controls for each of these signals: A and B.

A indicates the control register bit to be set to enable the corresponding signal to be output on the multiplexed pin.

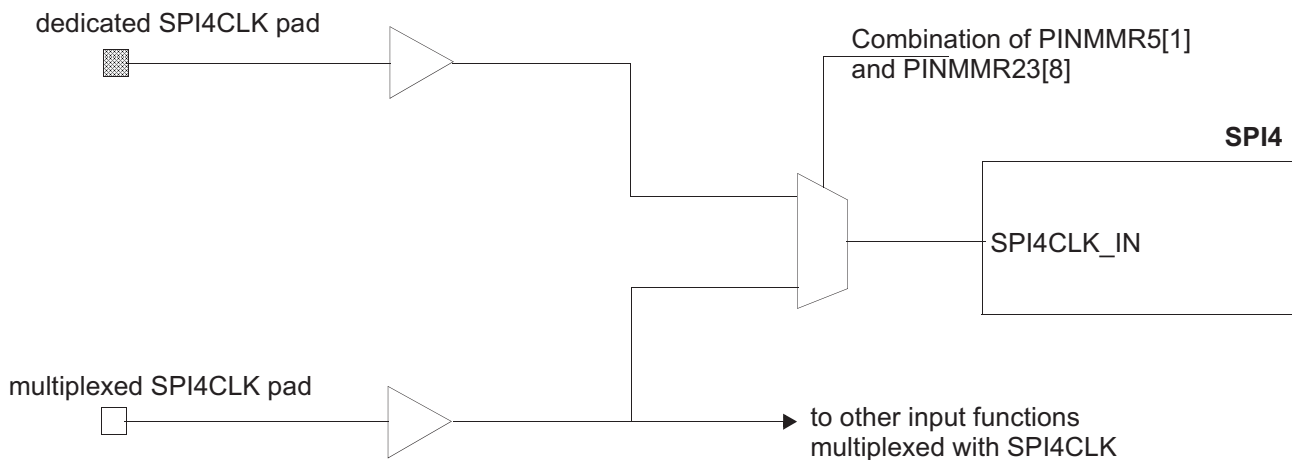
B is another control register bit that is used to select between the dedicated input terminal and the multiplexed pin.

- The input to a module comes from the **dedicated** pad when the condition [ **not(A) AND B** ] = TRUE.
- The input to a module comes from the **multiplexed** pin when the condition [ **A OR not(B)** ] = TRUE.

**NOTE: Inputs are broadcast to all multiplexed functions**

The input signals are broadcast to all modules hooked up to a terminal. The application must ensure that modules that are not being used in the application do not react to a change on their input functions. e.g. a GIO signal toggle can trigger an interrupt request, when the application actually is using the function multiplexed with this GIO signal.

**Figure 4-3. Input Multiplexing Example**



**4.3.3 Control of Special Multiplexed Options**

PINMMR29 and PINMMR30 registers are used to control some specific device functions.

• **EMIF\_CLK Control:**

PINMMR29[8] is set by default. This is used to block the EMIF\_CLK from being output from the microcontroller. If the EMIF is used to connect to an external SDRAM module, then the application must enable the EMIF\_CLK output by clearing the PINMMR29[8] bit.

• **Ethernet Controller Mode Control:**

PINMMR29[24] is set by default. This bit is used to enable the Reduced Media Independent Interface of the Ethernet controller. If the application desires to use the Media Independent Interface of the Ethernet controller, then the PINMMR29[24] must be cleared.

• **ADC Trigger Control:**

The microcontrollers contain two Analog-to-Digital Converter (ADC) modules. The ADC conversions can be started using a rising or falling or both edges as the trigger event. Both the ADC modules support up to eight event trigger inputs. There are two sets of these 8 inputs for each ADC. The first set is the default set and is selected by setting the PINMMR30[0]. The PINMMR30[0] is also set by default. The alternate set of 8 event trigger options are selected by clearing PINMMR30[0] bit.



## 4.4 Safety Features

The IOMM supports certain safety functions that are designed to prevent unintentional changes to the I/O multiplexing configuration. These are described in the following sections.

### 4.4.1 Locking Mechanism for Memory-Mapped Registers

The IOMM contains a mechanism to prevent any spurious writes from changing any of the PINMMR values. The PINMMRs are locked by default and after any system reset. None of the PINMMRs can be written under this condition. The application can read any of the IOMM registers regardless of the state of the locking mechanism.

- **Enabling Write Access to the PINMMRs**

To enable write access to the PINMMRs, the CPU must write 0x83e70b13 to the kick0 register followed by a write of 0x95a4f1e0 to the kick1 register.

- **Disabling Write Access to the PINMMRs**

It is recommended to disable write access to the PINMMRs once the I/O multiplexing configuration is completed. This can be done by:

- writing any other data value to either of the kick registers, or
- restarting the unlock sequence by writing 0x83e70b13 to the kick0 register

---

**NOTE: No Error On Write to Locked PINMMRs**

There is no error response on any write accesses to the PINMMRs when write access is disabled. None of the PINMMRs change state due to this write.

---

### 4.4.2 Master ID Check

Each bus transaction to the IOMM has an associated master ID value. Master ID values are unique to each bus master. The IOMM only allows write accessed by the CPU bus master. Writes by other bus masters will generate a bus error response and the states of the IOMM registers are not affected.

### 4.4.3 Error Conditions

The IOMM generates one error signal that is mapped to the Error Signaling Module's Group 1, channel 37. This error signal is generated under either of the following two conditions:

- Address Error – occurs when there is a read or a write access to an un-implemented memory location within the IOMM register frame.
- Protection Error – occurs when the CPU writes to an IOMM register while not in a privileged mode of operation.

## 4.5 IOMM Registers

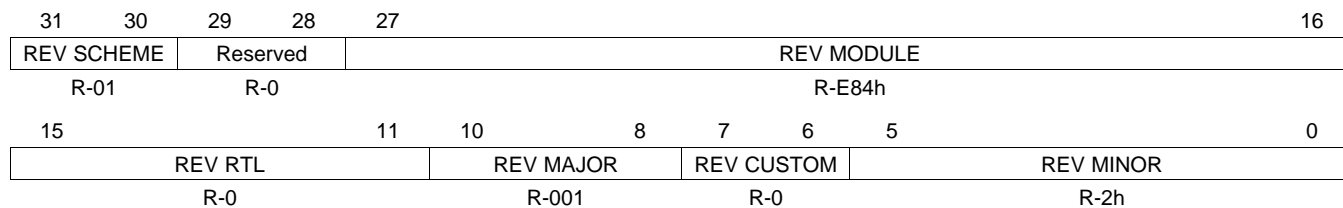
Table 4-2 shows a summary of the control registers in the IOMM.

**Table 4-2. IOMM Register Summary**

Offset	Acronym	Register Description	Section
00	REVISION_REG	Revision Register	<a href="#">Section 4.5.1</a>
20h	BOOT_REG	Boot Mode Register	<a href="#">Section 4.5.2</a>
38h	KICK_REG0	Kicker Register 0	<a href="#">Section 4.5.3</a>
3Ch	KICK_REG1	Kicker Register 1	<a href="#">Section 4.5.4</a>
E0h	ERR_RAW_STATUS_REG	Error Raw Status / Set Register	<a href="#">Section 4.5.5</a>
E4h	ERR_ENABLED_STATUS_REG	Error Enabled Status / Clear Register	<a href="#">Section 4.5.6</a>
E8h	ERR_ENABLE_REG	Error Signaling Enable Register	<a href="#">Section 4.5.7</a>
ECh	ERR_ENABLE_CLR_REG	Error Signaling Enable Clear Register	<a href="#">Section 4.5.8</a>
F4h	FAULT_ADDRESS_REG	Fault Address Register	<a href="#">Section 4.5.9</a>
F8h	FAULT_STATUS_REG	Fault Status Register	<a href="#">Section 4.5.10</a>
FCh	FAULT_CLEAR_REG	Fault Clear Register	<a href="#">Section 4.5.11</a>
110h - 188h	PINMMRnn	Pin Multiplexing Control Registers	<a href="#">Section 4.5.12</a>

### 4.5.1 REVISION\_REG: Revision Register

**Figure 4-4. REVISION\_REG: Revision Register (Offset = 00)**

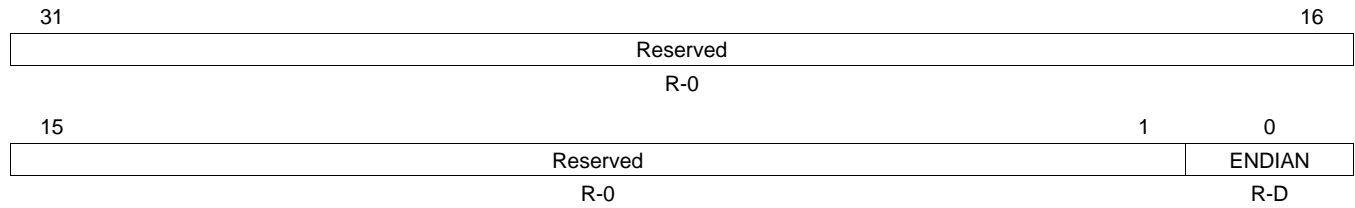


LEGEND: R/W = Read/Write; R = Read only; C = Clear; -n = value after reset

**Table 4-3. Revision Register Field Descriptions**

Bit	Field	Value	Description
31-30	REV SCHEME	01	Revision Scheme
29-28	Reserved	0	Reads return zeros, writes have no effect.
27-16	REV MODULE	E84h	Module Id
15-11	REV RTL	0	RTL Revision
10-8	REV MAJOR	001	Major Revision
7-6	REV CUSTOM	0	Custom Revision
5-0	REV MINOR	2h	Minor Revision

### 4.5.2 BOOT\_REG: Boot Mode Register

**Figure 4-5. BOOT\_REG: Boot Mode Register (Offset = 20h)**


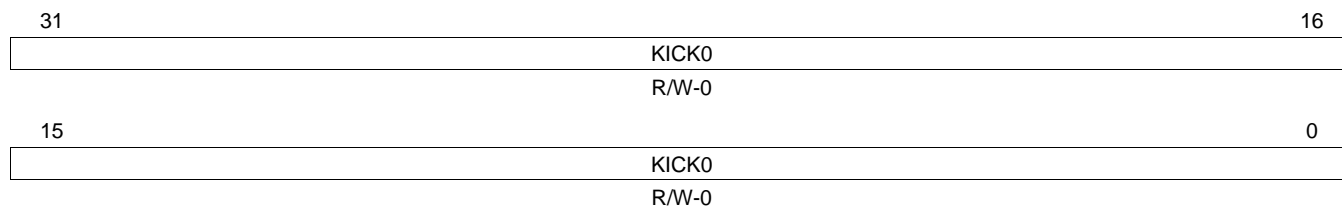
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset; R-D = Value read is determined by external configuration

**Table 4-4. Boot Mode Register Field Descriptions**

Bit	Field	Value	Description
31-1	Reserved	0	Reads return zeros, writes have no effect.
0	ENDIAN	0	Device endianness Device is configured in little endian mode
		1	Device is configured in big endian mode

### 4.5.3 KICK\_REG0: Kicker Register 0

**Figure 4-6. KICK\_REG0: Kicker Register 0 (Offset = 38h)**



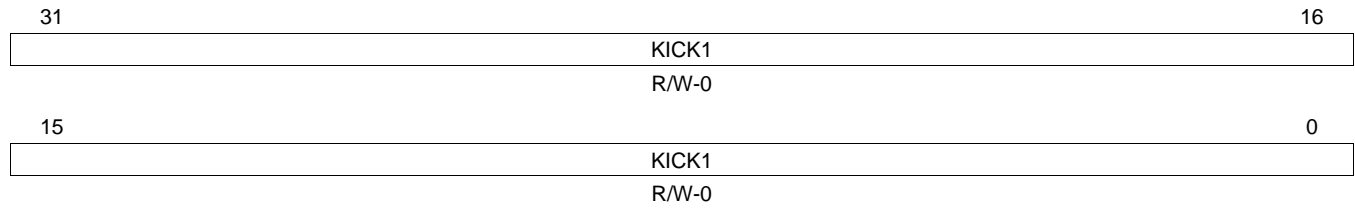
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 4-5. Kicker Register 0 Field Descriptions**

Bit	Field	Value	Description
31-0	KICK0	0	Kicker 0 Register. The value 83E7 0B13h must be written to KICK0 as part of the process to unlock the CPU write access to the PINMMRnn registers.

#### 4.5.4 KICK\_REG1: Kicker Register 1

**Figure 4-7. KICK\_REG1: Kicker Register 1 (Offset = 3Ch)**



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

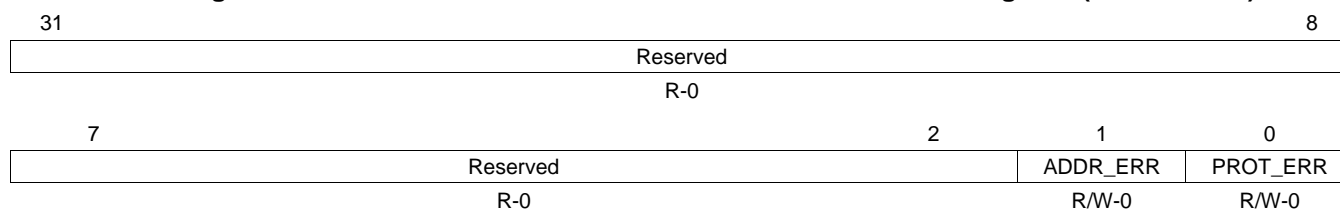
**Table 4-6. Kicker Register 1 Field Descriptions**

Bit	Field	Value	Description
31-0	KICK1	0	Kicker 1 Register. The value 95A4 F1E0h must be written to the KICK1 as part of the process to unlock the CPU write access to the PINMMRnn registers.

#### 4.5.5 ERR\_RAW\_STATUS\_REG: Error Raw Status / Set Register

This register shows the statuses of the error conditions (before enabling) and allows setting the status.

**Figure 4-8. ERR\_RAW\_STATUS\_REG: Error Raw Status / Set Register (Offset = E0h)**



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

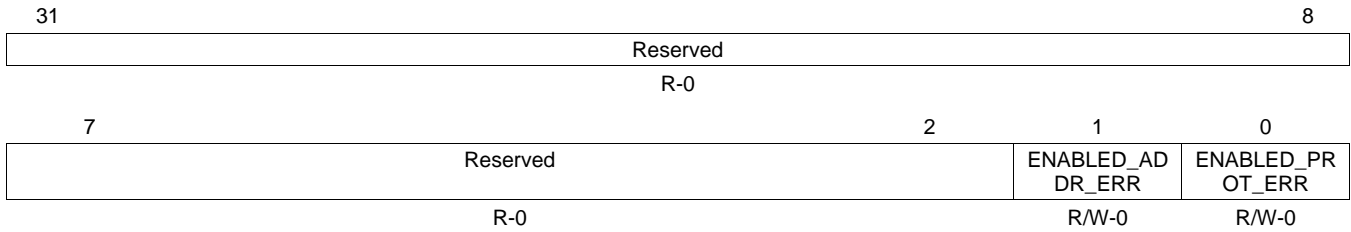
**Table 4-7. Error Raw Status / Set Register Field Descriptions**

Bit	Field	Value	Description
31-2	Reserved	0	Read returns zeros, writes have no effect.
1	ADDR_ERR	0	Addressing Error Status and Error Signaling Enable Read: Addressing Error has not occurred. Write: Writing 0 has no effect.
		1	Read: Addressing Error has been detected. Write: Addressing Error status is set.
0	PROT_ERR	0	Protection Error Status and Error Signaling Enable Read: Protection Error has not occurred. Write: Writing 0 has no effect.
		1	Read: Protection Error has been detected. Write: Protection Error status is set.

#### 4.5.6 ERR\_ENABLED\_STATUS\_REG: Error Enabled Status / Clear Register

This register shows the error signal enabled status and allows clearing of the error status.

**Figure 4-9. ERR\_ENABLED\_STATUS\_REG: Error Enabled Status / Clear Register (Offset = E4h)**



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

**Table 4-8. Error Signaling Enabled Status / Clear Register Field Descriptions**

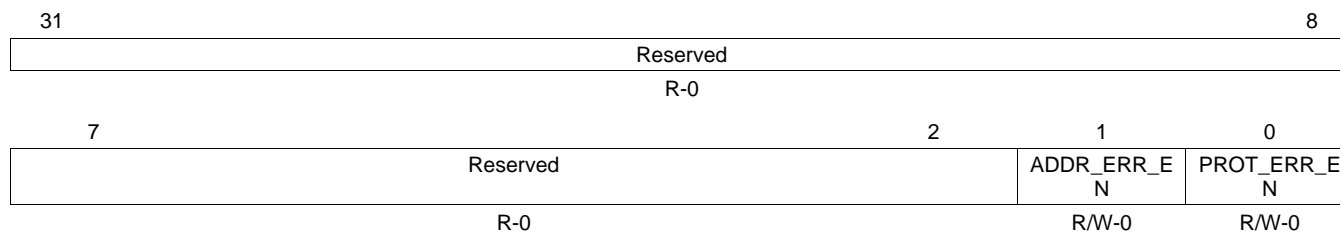
Bit	Field	Value	Description
31-2	Reserved	0	Read returns zeros, writes have no effect.
1	ENABLED_A DDR_ERR	0	Addressing Error Signaling Enable Status and Status Clear Read: Addressing Error Signaling is disabled. Write: Writing 0 has no effect.
		1	Read: Addressing Error Signaling is enabled. Write: Addressing Error status is cleared.
0	ENABLED_P ROT_ERR	0	Protection Error Signaling Enable Status and Status Clear Read: Protection Error Signaling is disabled. Write: Writing 0 has no effect.
		1	Read: Protection Error Signaling is enabled. Write: Protection Error status is cleared.



### 4.5.7 ERR\_ENABLE\_REG: Error Signaling Enable Register

This register shows the interrupt enable status and allows enabling of the interrupts.

**Figure 4-10. ERR\_ENABLE\_REG: Error Signaling Enable Register (Offset = E8h)**



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

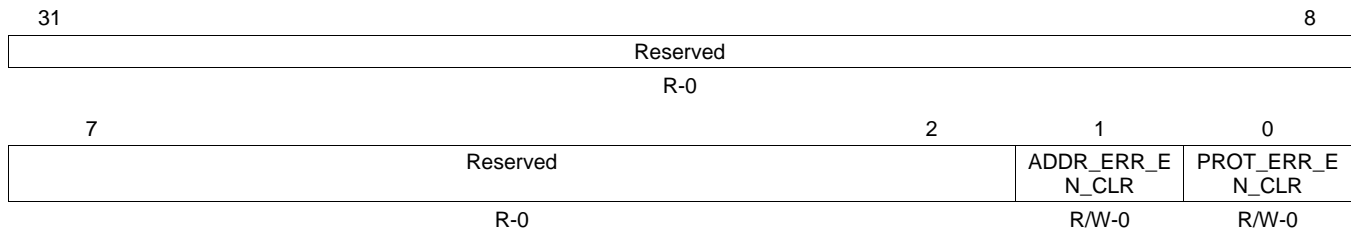
**Table 4-9. Error Enable Register Field Descriptions**

Bit	Field	Value	Description
31-2	Reserved	0	Read returns zeros, writes have no effect.
1	ADDR_ERR_EN	0	Addressing Error Signaling Enable Read: Addressing Error Signaling is disabled. Write: Writing 0 has no effect.
		1	Read: Addressing Error Signaling is enabled. Write: Addressing Error Signaling is enabled.
0	PROT_ERR_EN	0	Protection Error Signaling Enable Read: Protection Error Signaling is disabled. Write: Writing 0 has no effect.
		1	Read: Protection Error Signaling is enabled. Write: Protection Error Signaling is enabled.

#### 4.5.8 ERR\_ENABLE\_CLR\_REG: Error Signaling Enable Clear Register

This register shows the error signaling enable status and allows disabling of the error signaling.

**Figure 4-11. ERR\_ENABLE\_CLR\_REG: Error Signaling Enable Clear Register (Offset = ECh)**



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

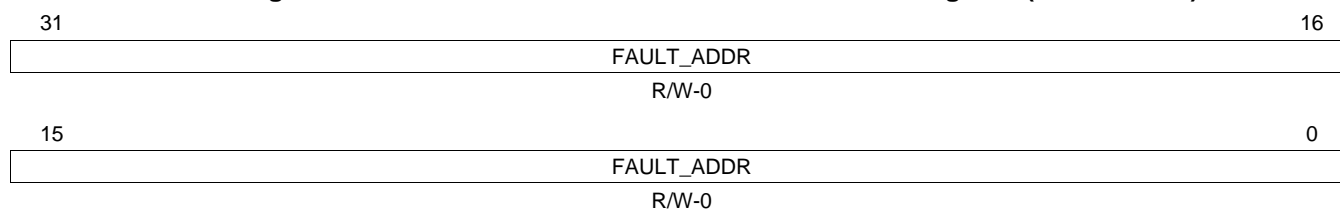
**Table 4-10. Interrupt Enable Clear Register Field Descriptions**

Bit	Field	Value	Description
31-2	Reserved	0	Read returns zeros, writes have no effect.
1	ADDR_ERR_EN_CLR	0	Addressing Error Signaling Enable Clear Read: Addressing Error signaling is disabled. Write: Writing 0 has no effect.
		1	Read: Addressing Error signaling is enabled. Write: Addressing Error signaling is disabled.
0	PROT_ERR_EN_CLR	0	Protection Error Signaling Enable Clear Read: Protection Error signaling is disabled. Write: Writing 0 has no effect.
		1	Read: Protection Error signaling is enabled. Write: Protection Error signaling is disabled.

#### 4.5.9 FAULT\_ADDRESS\_REG: Fault Address Register

This register holds the address of the first fault transfer.

**Figure 4-12. FAULT\_ADDRESS\_REG: Fault Address Register (Offset = F4h)**



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

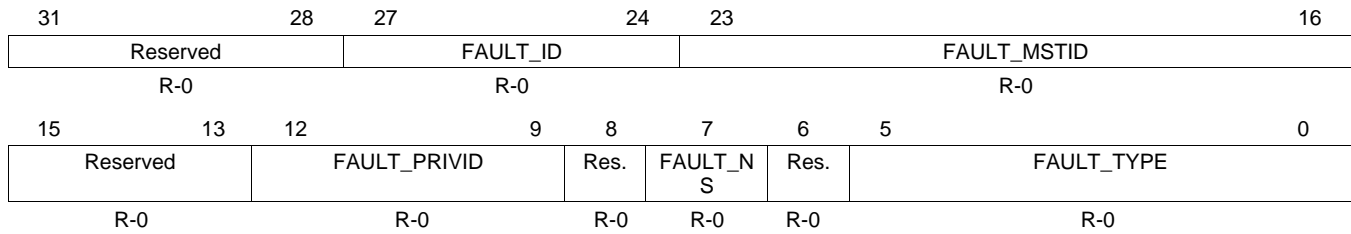
**Table 4-11. Fault Address Register Field Descriptions**

Bit	Field	Value	Description
31-0	FAULT_ADDR	0	Fault Address. This field captures the fault address in case of an address error or a protection error condition.

#### 4.5.10 FAULT\_STATUS\_REG: Fault Status Register

This register holds the status and attributes of the first fault transfer.

**Figure 4-13. FAULT\_STATUS\_REG: Fault Status Register (Offset = F8h)**



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

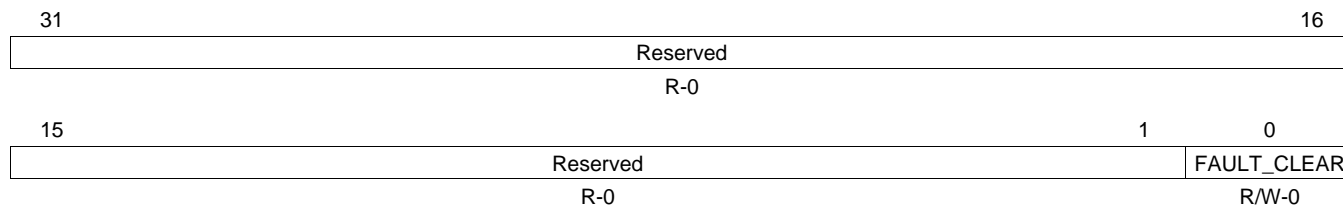
**Table 4-12. Fault Status Register Field Descriptions**

Bit	Field	Value	Description
31-28	Reserved	0	Reads return zeros, writes have no effect.
27-24	FAULT_ID		Faulting Transaction ID
23-16	FAULT_MSTID		ID of Master that initiated the faulting transaction
15-13	Reserved	0	Reads return zeros, writes have no effect.
12-9	FAULT_PRIVID		Faulting Privilege ID
8	Reserved	0	Reads return zeros, writes have no effect.
7	FAULT_NS		Fault: Non-secure access detected
6	Reserved	0	Reads return zeros, writes have no effect.
5-0	FAULT_TYPE		Type of fault detected
		20h	Supervisor read fault
		8h	Supervisor write fault
		4h	Supervisor execute fault
		3h	User read fault
		2h	User write fault
		1h	User execute fault
		0	No fault

#### 4.5.11 FAULT\_CLEAR\_REG: Fault Clear Register

This register allows the application to clear the current fault so that another can be captured when 1 is written to this register.

**Figure 4-14. FAULT\_CLEAR\_REG: Fault Clear Register (Offset = FCh )**



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

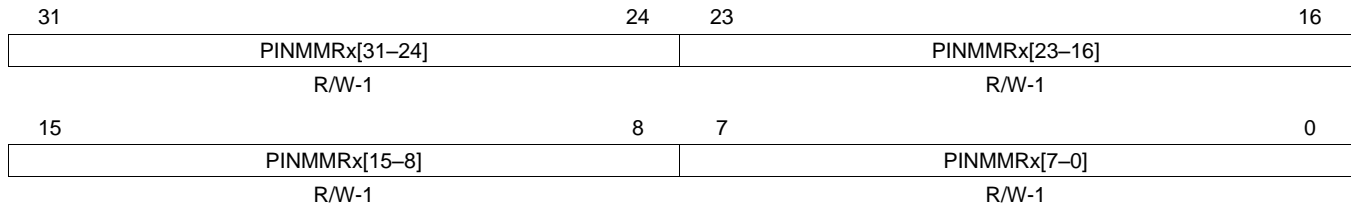
**Table 4-13. FAULT\_CLEAR\_REG: Fault Clear Register Field Descriptions**

Bit	Field	Value	Description
31-1	Reserved	0	Reads return zeros, writes have no effect.
0	FAULT_CLEAR	0	Fault Clear Read: Current value of the FAULT_CLEAR bit is 0. Write: Writing 0 has no effect.
		1	Read: Current value of the FAULT_CLEAR bit is 1. Write: Writing a 1 clears the current fault.

#### 4.5.12 PINMMRnn: Pin Multiplexing Control Registers

These registers control the multiplexing of the functionality available on each pad. There are 31 such registers – PINMMR0 through PINMMR31. Each 8-bit field of a PINMMR register controls the functionality of a single ball/pin. The mapping between the PINMMRx control registers and the functionality selected on a given terminal is defined in [Section 4.6](#).

**Figure 4-15. PINMMRnn: Pin Multiplexing Control Registers (Offset = 110h - 188h)**



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

**Table 4-14. Pin Multiplexing Control Registers Field Descriptions**

Bit	Field	Value	Description
31-24	PINMMRx[31–24]	1h	Each of these byte-fields control the functionality on a given ball/pin. Please refer to <a href="#">Table 4-15</a> for a list of multiplexed signals sorted by the control registers.
23-16	PINMMRx[23–16]	1h	
15-8	PINMMRx[15–8]	1h	
7-0	PINMMRx[7–0]	1h	

## 4.6 Signal Multiplexing and Control on Microcontrollers

**Table 4-15. Multiplexing and Control on Microcontrollers**

Default Function	Selection Bit	Alternate Function 1	Selection Bit	Alternate Function 2	Selection Bit	Alternate Function 3	Selection Bit	Alternate Function 4	Selection Bit
GIOB[3]	PINMMR0[0]	RESERVED	PINMMR0[1]	RESERVE D	PINMMR0[2]	RESERVED	PINMMR0[3]	RESERVE D	PINMMR0[4]
GIOA[0]	PINMMR0[8]	RESERVED	PINMMR0[9]	RESERVE D	PINMMR0[10]	RESERVED	PINMMR0[11]	RESERVE D	PINMMR0[12]
MIBSPI3NCS[3]	PINMMR0[16]	I2C_SCL	PINMMR0[17]	N2HET1[2 9]	PINMMR0[18]	RESERVED	PINMMR0[19]	RESERVE D	PINMMR0[20]
MIBSPI3NCS[2]	PINMMR0[24]	I2C_SDA	PINMMR0[25]	N2HET1[2 7]	PINMMR0[26]	RESERVED	PINMMR0[27]	RESERVE D	PINMMR0[28]
GIOA[1]	PINMMR1[0]	RESERVED	PINMMR1[1]	RESERVE D	PINMMR1[2]	RESERVED	PINMMR1[3]	RESERVE D	PINMMR1[4]
N2HET1[11]	PINMMR1[8]	MIBSPI3NCS[4]	PINMMR1[9]	N2HET2[1 8]	PINMMR1[10]	RESERVED	PINMMR1[11]	RESERVE D	PINMMR1[12]
ETMDATA[2 0]	PINMMR1[16]	EMIF_DATA[4]	PINMMR1[17]	RESERVE D	PINMMR1[18]	RESERVED	PINMMR1[19]	RESERVE D	PINMMR1[20]
ETMDATA[2 1]	PINMMR1[24]	EMIF_DATA[5]	PINMMR1[25]	RESERVE D	PINMMR1[26]	RESERVED	PINMMR1[27]	RESERVE D	PINMMR1[28]
GIOA[2]	PINMMR2[0]	RESERVED	PINMMR2[1]	RESERVE D	PINMMR2[2]	N2HET2[0]	PINMMR2[3]	RESERVE D	PINMMR2[4]
ETMDATA[2 2]	PINMMR2[8]	EMIF_DATA[6]	PINMMR2[9]	RESERVE D	PINMMR2[10]	RESERVED	PINMMR2[11]	RESERVE D	PINMMR2[12]
GIOA[3]	PINMMR2[16]	N2HET2[2]	PINMMR2[17]	RESERVE D	PINMMR2[18]	RESERVED	PINMMR2[19]	RESERVE D	PINMMR2[20]
GIOA[5]	PINMMR2[24]	EXTCLKIN	PINMMR2[25]	RESERVE D	PINMMR2[26]	RESERVED	PINMMR2[27]	RESERVE D	PINMMR2[28]
ETMDATA[2 3]	PINMMR3[0]	EMIF_DATA[7]	PINMMR3[1]	RESERVE D	PINMMR3[2]	RESERVED	PINMMR3[3]	RESERVE D	PINMMR3[4]
N2HET1[22]	PINMMR3[8]	RESERVED	PINMMR3[9]	RESERVE D	PINMMR3[10]	RESERVED	PINMMR3[11]	RESERVE D	PINMMR3[12]
GIOA[6]	PINMMR3[16]	N2HET2[4]	PINMMR3[17]	RESERVE D	PINMMR3[18]	RESERVED	PINMMR3[19]	RESERVE D	PINMMR3[20]
ETMDATA[2 4]	PINMMR3[24]	EMIF_DATA[8]	PINMMR3[25]	RESERVE D	PINMMR3[26]	RESERVED	PINMMR3[27]	RESERVE D	PINMMR3[28]
GIOA[7]	PINMMR4[0]	N2HET2[6]	PINMMR4[1]	RESERVE D	PINMMR4[2]	RESERVED	PINMMR4[3]	RESERVE D	PINMMR4[4]
ETMDATA[2 5]	PINMMR4[8]	EMIF_DATA[9]	PINMMR4[9]	RESERVE D	PINMMR4[10]	RESERVED	PINMMR4[11]	RESERVE D	PINMMR4[12]
N2HET1[01]	PINMMR4[16]	SPI4NENA	PINMMR4[17]	RESERVE D	PINMMR4[18]	RESERVED	PINMMR4[19]	N2HET2[8]	PINMMR4[20]
N2HET1[03]	PINMMR4[24]	SPI4NCS[0]	PINMMR4[25]	RESERVE D	PINMMR4[26]	RESERVED	PINMMR4[27]	N2HET2[1 0]	PINMMR4[28]
N2HET1[0]	PINMMR5[0]	SPI4CLK	PINMMR5[1]	RESERVE D	PINMMR5[2]	RESERVED	PINMMR5[3]	RESERVE D	PINMMR5[4]
N2HET1[02]	PINMMR5[8]	SPI4SIMO	PINMMR5[9]	RESERVE D	PINMMR5[10]	RESERVED	PINMMR5[11]	RESERVE D	PINMMR5[12]
N2HET1[05]	PINMMR5[16]	SPI4SOMI	PINMMR5[17]	N2HET2[1 2]	PINMMR5[18]	RESERVED	PINMMR5[19]	RESERVE D	PINMMR5[20]
ETMDATA[2 6]	PINMMR5[24]	EMIF_DATA[10]	PINMMR5[25]	RESERVE D	PINMMR5[26]	RESERVED	PINMMR5[27]	RESERVE D	PINMMR5[28]
N2HET1[07]	PINMMR6[0]	RESERVED	PINMMR6[1]	RESERVE D	PINMMR6[2]	N2HET2[14]	PINMMR6[3]	RESERVE D	PINMMR6[4]
ETMDATA[2 7]	PINMMR6[8]	EMIF_DATA[11]	PINMMR6[9]	RESERVE D	PINMMR6[10]	RESERVED	PINMMR6[11]	RESERVE D	PINMMR6[12]

**Table 4-15. Multiplexing and Control on Microcontrollers (continued)**

Default Function	Selection Bit	Alternate Function 1	Selection Bit	Alternate Function 2	Selection Bit	Alternate Function 3	Selection Bit	Alternate Function 4	Selection Bit
N2HET1[09]	PINMMR6[16]	N2HET2[16]	PINMMR6[17]	RESERVED	PINMMR6[18]	RESERVED	PINMMR6[19]	RESERVED	PINMMR6[20]
ETMDATA[28]	PINMMR6[24]	EMIF_DATA[12]	PINMMR6[25]	RESERVED	PINMMR6[26]	RESERVED	PINMMR6[27]	RESERVED	PINMMR6[28]
ETMDATA[29]	PINMMR7[0]	EMIF_DATA[13]	PINMMR7[1]	RESERVED	PINMMR7[2]	RESERVED	PINMMR7[3]	RESERVED	PINMMR7[4]
MIBSPI3NCS[1]	PINMMR7[8]	N2HET1[25]	PINMMR7[9]	MDCLK	PINMMR7[10]	RESERVED	PINMMR7[11]	RESERVED	PINMMR7[12]
N2HET1[06]	PINMMR7[16]	SCIRX	PINMMR7[17]	RESERVED	PINMMR7[18]	RESERVED	PINMMR7[19]	RESERVED	PINMMR7[20]
ETMDATA[30]	PINMMR7[24]	EMIF_DATA[14]	PINMMR7[25]	RESERVED	PINMMR7[26]	RESERVED	PINMMR7[27]	RESERVED	PINMMR7[28]
N2HET1[13]	PINMMR8[0]	SCITX	PINMMR8[1]	RESERVED	PINMMR8[2]	RESERVED	PINMMR8[3]	RESERVED	PINMMR8[4]
MIBSPI1NCS[2]	PINMMR8[8]	N2HET1[19]	PINMMR8[9]	MDIO	PINMMR8[10]	RESERVED	PINMMR8[11]	RESERVED	PINMMR8[12]
N2HET1[15]	PINMMR8[16]	MIBSPI1NCS[4]	PINMMR8[17]	RESERVED	PINMMR8[18]	RESERVED	PINMMR8[19]	RESERVED	PINMMR8[20]
ETMDATA[31]	PINMMR8[24]	EMIF_DATA[15]	PINMMR8[25]	RESERVED	PINMMR8[26]	RESERVED	PINMMR8[27]	RESERVED	PINMMR8[28]
ETMTRACE CLKIN	PINMMR9[0]	EXTCLKIN2	PINMMR9[1]	RESERVED	PINMMR9[2]	RESERVED	PINMMR9[3]	RESERVED	PINMMR9[4]
MIBSPI3NENA	PINMMR9[8]	MIBSPI3NCS[5]	PINMMR9[9]	N2HET1[31]	PINMMR9[10]	RESERVED	PINMMR9[11]	RESERVED	PINMMR9[12]
MIBSPI3NCS[0]	PINMMR9[16]	AD2EVT	PINMMR9[17]	GIOB[2]	PINMMR2[9][16]	RESERVED	PINMMR9[19]	RESERVED	PINMMR9[20]
MIBSPI1NCS[3]	PINMMR9[24]	N2HET1[21]	PINMMR9[25]	RESERVED	PINMMR9[26]	RESERVED	PINMMR9[27]	RESERVED	PINMMR9[28]
AD1EVT	PINMMR10[0]	MII_RX_ER	PINMMR10[1]	RMII_RX_ER	PINMMR10[2]	RESERVED	PINMMR10[3]	RESERVED	PINMMR10[4]
ETMDATA[19]	PINMMR10[8]	EMIF_DATA[3]	PINMMR10[9]	RESERVED	PINMMR10[10]	RESERVED	PINMMR10[11]	RESERVED	PINMMR10[12]
EMIF_nCS[0]	PINMMR10[16]	RTP_DATA[15]	PINMMR10[17]	N2HET2[7]	PINMMR10[18]	RESERVED	PINMMR10[19]	RESERVED	PINMMR10[20]
ETMDATA[18]	PINMMR10[24]	EMIF_DATA[2]	PINMMR10[25]	RESERVED	PINMMR10[26]	RESERVED	PINMMR10[27]	RESERVED	PINMMR10[28]
EMIF_nCS[3]	PINMMR11[0]	RTP_DATA[14]	PINMMR11[1]	N2HET2[9]	PINMMR11[2]	RESERVED	PINMMR11[3]	RESERVED	PINMMR11[4]
EMIF_nCS[4]	PINMMR11[8]	RTP_DATA[07]	PINMMR11[9]	RESERVED	PINMMR11[10]	RESERVED	PINMMR11[11]	RESERVED	PINMMR11[12]
ETMDATA[17]	PINMMR11[16]	EMIF_DATA[1]	PINMMR11[17]	RESERVED	PINMMR11[18]	RESERVED	PINMMR11[19]	RESERVED	PINMMR11[20]
N2HET1[24]	PINMMR11[24]	MIBSPI1NCS[5]	PINMMR11[25]	MII_RXD[0]	PINMMR11[26]	RESERVED	PINMMR11[27]	RESERVED	PINMMR11[28]
N2HET1[26]	PINMMR12[0]	MII_RXD[1]	PINMMR12[1]	RMII_RXD[1]	PINMMR12[2]	RESERVED	PINMMR12[3]	RESERVED	PINMMR12[4]
ETMDATA[16]	PINMMR12[8]	EMIF_DATA[0]	PINMMR12[9]	RESERVED	PINMMR12[10]	RESERVED	PINMMR12[11]	RESERVED	PINMMR12[12]
MIBSPI1NENA	PINMMR12[16]	N2HET1[23]	PINMMR12[17]	MII_RXD[2]	PINMMR12[18]	RESERVED	PINMMR12[19]	RESERVED	PINMMR12[20]
MIBSPI5NENA	PINMMR12[24]	DMM_DATA[7]	PINMMR12[25]	MII_RXD[3]	PINMMR12[26]	RESERVED	PINMMR12[27]	RESERVED	PINMMR12[28]
MIBSPI5SO MI[0]	PINMMR13[0]	DMM_DATA[12]	PINMMR13[1]	MII_TXD[0]	PINMMR13[2]	RESERVED	PINMMR13[3]	RESERVED	PINMMR13[4]



**Table 4-15. Multiplexing and Control on Microcontrollers (continued)**

Default Function	Selection Bit	Alternate Function 1	Selection Bit	Alternate Function 2	Selection Bit	Alternate Function 3	Selection Bit	Alternate Function 4	Selection Bit
MIBSPI5SIMO[0]	PINMMR1 3[8]	DMM_DAT A[8]	PINMMR1 3[9]	MII_TXD[1 ]	PINMMR1 3[10]	RMII_TXD[1 ]	PINMMR1 3[11]	RESERVE D	PINMMR1 3[12]
MIBSPI5CLK	PINMMR1 3[16]	DMM_DAT A[4]	PINMMR1 3[17]	MII_TXEN	PINMMR1 3[18]	RMII_TXEN	PINMMR1 3[19]	RESERVE D	PINMMR1 3[20]
MIBSPI1NCS[0]	PINMMR1 3[24]	MIBSPI1SO MI[1]	PINMMR1 3[25]	MII_TXD[2 ]	PINMMR1 3[26]	RESERVED	PINMMR1 3[27]	RESERVE D	PINMMR1 3[28]
N2HET1[8]	PINMMR1 4[0]	MIBSPI1SI MO[1]	PINMMR1 4[1]	MII_TXD[3 ]	PINMMR1 4[2]	RESERVED	PINMMR1 4[3]	RESERVE D	PINMMR1 4[4]
N2HET1[28]	PINMMR1 4[8]	MII_RXCLK	PINMMR1 4[9]	RMII_REF CLK	PINMMR1 4[10]	MII_RX_AV CLK4	PINMMR1 4[11]	RESERVE D	PINMMR1 4[12]
EMIF_nWE	PINMMR1 4[16]	EMIF_RNW	PINMMR1 4[17]	RESERVE D	PINMMR1 4[18]	RESERVED	PINMMR1 4[19]	RESERVE D	PINMMR1 4[20]
EMIF_BA[1]	PINMMR1 4[24]	N2HET2[5]	PINMMR1 4[25]	RESERVE D	PINMMR1 4[26]	RESERVED	PINMMR1 4[27]	RESERVE D	PINMMR1 4[28]
EMIF_ADDR[21]	PINMMR1 5[0]	RTP_CLK	PINMMR1 5[1]	RESERVE D	PINMMR1 5[2]	RESERVED	PINMMR1 5[3]	RESERVE D	PINMMR1 5[4]
EMIF_ADDR[20]	PINMMR1 5[8]	RTP_nSYN C	PINMMR1 5[9]	RESERVE D	PINMMR1 5[10]	RESERVED	PINMMR1 5[11]	RESERVE D	PINMMR1 5[12]
EMIF_ADDR[19]	PINMMR1 5[16]	RTP_nENA	PINMMR1 5[17]	RESERVE D	PINMMR1 5[18]	RESERVED	PINMMR1 5[19]	RESERVE D	PINMMR1 5[20]
EMIF_ADDR[18]	PINMMR1 5[24]	RTP_DATA[0]	PINMMR1 5[25]	RESERVE D	PINMMR1 5[26]	RESERVED	PINMMR1 5[27]	RESERVE D	PINMMR1 5[28]
ETMDATA[12]	PINMMR1 6[0]	EMIF_BA[0]	PINMMR1 6[1]	RESERVE D	PINMMR1 6[2]	RESERVED	PINMMR1 6[3]	RESERVE D	PINMMR1 6[4]
EMIF_ADDR[17]	PINMMR1 6[8]	RTP_DATA[01]	PINMMR1 6[9]	RESERVE D	PINMMR1 6[10]	RESERVED	PINMMR1 6[11]	RESERVE D	PINMMR1 6[12]
EMIF_ADDR[16]	PINMMR1 6[16]	RTP_DATA[02]	PINMMR1 6[17]	RESERVE D	PINMMR1 6[18]	RESERVED	PINMMR1 6[19]	RESERVE D	PINMMR1 6[20]
ETMDATA[13]	PINMMR1 6[24]	EMIF_nOE	PINMMR1 6[25]	RESERVE D	PINMMR1 6[26]	RESERVED	PINMMR1 6[27]	RESERVE D	PINMMR1 6[28]
N2HET1[10]	PINMMR1 7[0]	MII_TX_CLK	PINMMR1 7[1]	RESERVE D	PINMMR1 7[2]	MII_TX_AV CLK4	PINMMR1 7[3]	RESERVE D	PINMMR1 7[4]
ETMDATA[14]	PINMMR1 7[8]	EMIF_nDQ M[1]	PINMMR1 7[9]	RESERVE D	PINMMR1 7[10]	RESERVED	PINMMR1 7[11]	RESERVE D	PINMMR1 7[12]
N2HET1[12]	PINMMR1 7[16]	MII_CRCS	PINMMR1 7[17]	RMII_CRCS_DV	PINMMR1 7[18]	RESERVED	PINMMR1 7[19]	RESERVE D	PINMMR1 7[20]
ETMDATA[8]	PINMMR1 7[24]	EMIF_ADD R[5]	PINMMR1 7[25]	RESERVE D	PINMMR1 7[26]	RESERVED	PINMMR1 7[27]	RESERVE D	PINMMR1 7[28]
EMIF_ADDR[15]	PINMMR1 8[0]	RTP_DATA[03]	PINMMR1 8[1]	RESERVE D	PINMMR1 8[2]	RESERVED	PINMMR1 8[3]	RESERVE D	PINMMR1 8[4]
N2HET1[14]	PINMMR1 8[8]	RESERVED	PINMMR1 8[9]	RESERVE D	PINMMR1 8[10]	RESERVED	PINMMR1 8[11]	RESERVE D	PINMMR1 8[12]
EMIF_ADDR[14]	PINMMR1 8[16]	RTP_DATA[04]	PINMMR1 8[17]	RESERVE D	PINMMR1 8[18]	RESERVED	PINMMR1 8[19]	RESERVE D	PINMMR1 8[20]
GI0B[0]	PINMMR1 8[24]	RESERVED	PINMMR1 8[25]	RESERVE D	PINMMR1 8[26]	RESERVED	PINMMR1 8[27]	RESERVE D	PINMMR1 8[28]
ETMDATA[9]	PINMMR1 9[0]	EMIF_ADD R[4]	PINMMR1 9[1]	RESERVE D	PINMMR1 9[2]	RESERVED	PINMMR1 9[3]	RESERVE D	PINMMR1 9[4]
N2HET1[30]	PINMMR1 9[8]	MII_RX_DV	PINMMR1 9[9]	RESERVE D	PINMMR1 9[10]	RESERVED	PINMMR1 9[11]	RESERVE D	PINMMR1 9[12]
ETMDATA[15]	PINMMR1 9[16]	EMIF_nDQ M[0]	PINMMR1 9[17]	RESERVE D	PINMMR1 9[18]	RESERVED	PINMMR1 9[19]	RESERVE D	PINMMR1 9[20]
ETMDATA[10]	PINMMR1 9[24]	EMIF_ADD R[3]	PINMMR1 9[25]	RESERVE D	PINMMR1 9[26]	RESERVED	PINMMR1 9[27]	RESERVE D	PINMMR1 9[28]

**Table 4-15. Multiplexing and Control on Microcontrollers (continued)**

Default Function	Selection Bit	Alternate Function 1	Selection Bit	Alternate Function 2	Selection Bit	Alternate Function 3	Selection Bit	Alternate Function 4	Selection Bit
EMIF_ADDR[13]	PINMMR2_0[0]	RTP_DATA[05]	PINMMR2_0[1]	RESERVE_D	PINMMR2_0[2]	RESERVED	PINMMR2_0[3]	RESERVE_D	PINMMR2_0[4]
EMIF_ADDR[12]	PINMMR2_0[8]	RTP_DATA[06]	PINMMR2_0[9]	RESERVE_D	PINMMR2_0[10]	RESERVED	PINMMR2_0[11]	RESERVE_D	PINMMR2_0[12]
MIBSPI1NCS[1]	PINMMR2_0[16]	N2HET1[17]	PINMMR2_0[17]	MI_COL	PINMMR2_0[18]	RESERVED	PINMMR2_0[19]	RESERVE_D	PINMMR2_0[20]
EMIF_ADDR[11]	PINMMR2_0[24]	RTP_DATA[8]	PINMMR2_0[25]	RESERVE_D	PINMMR2_0[26]	RESERVED	PINMMR2_0[27]	RESERVE_D	PINMMR2_0[28]
EMIF_ADDR[1]	PINMMR2_1[0]	N2HET2[3]	PINMMR2_1[1]	RESERVE_D	PINMMR2_1[2]	RESERVED	PINMMR2_1[3]	RESERVE_D	PINMMR2_1[4]
GIOB[1]	PINMMR2_1[8]	RESERVED	PINMMR2_1[9]	RESERVE_D	PINMMR2_1[10]	RESERVED	PINMMR2_1[11]	RESERVE_D	PINMMR2_1[12]
EMIF_ADDR[10]	PINMMR2_1[16]	RTP_DATA[09]	PINMMR2_1[17]	RESERVE_D	PINMMR2_1[18]	RESERVED	PINMMR2_1[19]	RESERVE_D	PINMMR2_1[20]
EMIF_ADDR[9]	PINMMR2_1[24]	RTP_DATA[10]	PINMMR2_1[25]	RESERVE_D	PINMMR2_1[26]	RESERVED	PINMMR2_1[27]	RESERVE_D	PINMMR2_1[28]
EMIF_ADDR[0]	PINMMR2_2[0]	N2HET2[1]	PINMMR2_2[1]	RESERVE_D	PINMMR2_2[2]	RESERVED	PINMMR2_2[3]	RESERVE_D	PINMMR2_2[4]
EMIF_ADDR[7]	PINMMR2_2[8]	RTP_DATA[12]	PINMMR2_2[9]	N2HET2[13]	PINMMR2_2[10]	RESERVED	PINMMR2_2[11]	RESERVE_D	PINMMR2_2[12]
EMIF_ADDR[6]	PINMMR2_2[16]	RTP_DATA[13]	PINMMR2_2[17]	N2HET2[11]	PINMMR2_2[18]	RESERVED	PINMMR2_2[19]	RESERVE_D	PINMMR2_2[20]
ETMDATA[11]	PINMMR2_2[24]	EMIF_ADD_R[2]	PINMMR2_2[25]	RESERVE_D	PINMMR2_2[26]	RESERVED	PINMMR2_2[27]	RESERVE_D	PINMMR2_2[28]
EMIF_ADDR[8]	PINMMR2_3[0]	RTP_DATA[11]	PINMMR2_3[1]	N2HET2[15]	PINMMR2_3[2]	RESERVED	PINMMR2_3[3]	RESERVE_D	PINMMR2_3[4]
SPI4CLK	PINMMR2_3[8]	RESERVED	PINMMR2_3[9]	RESERVE_D	PINMMR2_3[10]	RESERVED	PINMMR2_3[11]	RESERVE_D	PINMMR2_3[12]
SPI4SIMO	PINMMR2_3[16]	RESERVED	PINMMR2_3[17]	RESERVE_D	PINMMR2_3[18]	RESERVED	PINMMR2_3[19]	RESERVE_D	PINMMR2_3[20]
SPI4SOMI	PINMMR2_3[24]	RESERVED	PINMMR2_3[25]	RESERVE_D	PINMMR2_3[26]	RESERVED	PINMMR2_3[27]	RESERVE_D	PINMMR2_3[28]
SPI4NENA	PINMMR2_4[0]	RESERVED	PINMMR2_4[1]	RESERVE_D	PINMMR2_4[2]	RESERVED	PINMMR2_4[3]	RESERVE_D	PINMMR2_4[4]
SPI4NCS[0]	PINMMR2_4[8]	RESERVED	PINMMR2_4[9]	RESERVE_D	PINMMR2_4[10]	RESERVED	PINMMR2_4[11]	RESERVE_D	PINMMR2_4[12]
N2HET1[17]	PINMMR2_4[16]	RESERVED	PINMMR2_4[17]	RESERVE_D	PINMMR2_4[18]	RESERVED	PINMMR2_4[19]	RESERVE_D	PINMMR2_4[20]
N2HET1[19]	PINMMR2_4[24]	RESERVED	PINMMR2_4[25]	RESERVE_D	PINMMR2_4[26]	RESERVED	PINMMR2_4[27]	RESERVE_D	PINMMR2_4[28]
N2HET1[21]	PINMMR2_5[0]	RESERVED	PINMMR2_5[1]	RESERVE_D	PINMMR2_5[2]	RESERVED	PINMMR2_5[3]	RESERVE_D	PINMMR2_5[4]
N2HET1[23]	PINMMR2_5[8]	RESERVED	PINMMR2_5[9]	RESERVE_D	PINMMR2_5[10]	RESERVED	PINMMR2_5[11]	RESERVE_D	PINMMR2_5[12]
N2HET1[25]	PINMMR2_5[16]	RESERVED	PINMMR2_5[17]	RESERVE_D	PINMMR2_5[18]	RESERVED	PINMMR2_5[19]	RESERVE_D	PINMMR2_5[20]
N2HET1[27]	PINMMR2_5[24]	RESERVED	PINMMR2_5[25]	RESERVE_D	PINMMR2_5[26]	RESERVED	PINMMR2_5[27]	RESERVE_D	PINMMR2_5[28]
N2HET1[29]	PINMMR2_6[0]	RESERVED	PINMMR2_6[1]	RESERVE_D	PINMMR2_6[2]	RESERVED	PINMMR2_6[3]	RESERVE_D	PINMMR2_6[4]
N2HET1[31]	PINMMR2_6[8]	RESERVED	PINMMR2_6[9]	RESERVE_D	PINMMR2_6[10]	RESERVED	PINMMR2_6[11]	RESERVE_D	PINMMR2_6[12]
MIBSPI5NCS[2]	PINMMR2_6[16]	DMM_DATA[2]	PINMMR2_6[17]	RESERVE_D	PINMMR2_6[18]	RESERVED	PINMMR2_6[19]	RESERVE_D	PINMMR2_6[20]

**Table 4-15. Multiplexing and Control on Microcontrollers (continued)**

Default Function	Selection Bit	Alternate Function 1	Selection Bit	Alternate Function 2	Selection Bit	Alternate Function 3	Selection Bit	Alternate Function 4	Selection Bit
MIBSPI5NCS [3]	PINMMR2 6[24]	DMM_DAT A[3]	PINMMR2 6[25]	RESERVE D	PINMMR2 6[26]	RESERVED	PINMMR2 6[27]	RESERVE D	PINMMR2 6[28]
MIBSPI5NCS [0]	PINMMR2 7[0]	DMM_DAT A[5]	PINMMR2 7[1]	RESERVE D	PINMMR2 7[2]	RESERVED	PINMMR2 7[3]	RESERVE D	PINMMR2 7[4]
MIBSPI5NCS [1]	PINMMR2 7[8]	DMM_DAT A[6]	PINMMR2 7[9]	RESERVE D	PINMMR2 7[10]	RESERVED	PINMMR2 7[11]	RESERVE D	PINMMR2 7[12]
MIBSPI5SIM O[1]	PINMMR2 7[16]	DMM_DAT A[9]	PINMMR2 7[17]	RESERVE D	PINMMR2 7[18]	RESERVED	PINMMR2 7[19]	RESERVE D	PINMMR2 7[20]
MIBSPI5SIM O[2]	PINMMR2 7[24]	DMM_DAT A[10]	PINMMR2 7[25]	RESERVE D	PINMMR2 7[26]	RESERVED	PINMMR2 7[27]	RESERVE D	PINMMR2 7[28]
MIBSPI5SIM O[3]	PINMMR2 8[0]	DMM_DAT A[11]	PINMMR2 8[1]	RESERVE D	PINMMR2 8[2]	RESERVED	PINMMR2 8[3]	RESERVE D	PINMMR2 8[4]
MIBSPI5SO MI[1]	PINMMR2 8[8]	DMM_DAT A[13]	PINMMR2 8[9]	RESERVE D	PINMMR2 8[10]	RESERVED	PINMMR2 8[11]	RESERVE D	PINMMR2 8[12]
MIBSPI5SO MI[2]	PINMMR2 8[16]	DMM_DAT A[14]	PINMMR2 8[17]	RESERVE D	PINMMR2 8[18]	RESERVED	PINMMR2 8[19]	RESERVE D	PINMMR2 8[20]
MIBSPI5SO MI[3]	PINMMR2 8[24]	DMM_DAT A[15]	PINMMR2 8[25]	RESERVE D	PINMMR2 8[26]	RESERVED	PINMMR2 8[27]	RESERVE D	PINMMR2 8[28]
SPI2NENA	PINMMR2 9[0]	SPI2NCS[1]	PINMMR2 9[1]	RESERVE D	PINMMR2 9[2]	RESERVED	PINMMR2 9[3]	RESERVE D	PINMMR2 9[4]
EMIF_CLK_SEL	PINMMR2 9[8]	RESERVED	PINMMR2 9[9]	RESERVE D	PINMMR2 9[10]	RESERVED	PINMMR2 9[11]	RESERVE D	PINMMR2 9[12]
GIOB[2]	PINMMR2 9[16]	RESERVED	PINMMR2 9[17]	RESERVE D	PINMMR2 9[18]	RESERVED	PINMMR2 9[19]	RESERVE D	PINMMR2 9[20]
GMII_SEL	PINMMR2 9[24]	RESERVED	PINMMR2 9[25]	RESERVE D	PINMMR2 9[26]	RESERVED	PINMMR2 9[27]	RESERVE D	PINMMR2 9[28]
ADC_TRG1	PINMMR3 0[0]	ADC_TRG2	PINMMR3 0[1]	RESERVE D	PINMMR3 0[2]	RESERVED	PINMMR3 0[3]	RESERVE D	PINMMR3 0[4]

## **F021 Flash Module**

---

---

---

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 chapter describes the F021 Flash module.

<b>Topic</b>	<b>Page</b>
<b>5.1 Overview .....</b>	<b>274</b>
<b>5.2 Default Flash Configuration .....</b>	<b>276</b>
<b>5.3 Memory Map .....</b>	<b>276</b>
<b>5.4 Control Registers .....</b>	<b>277</b>

## 5.1 Overview

The F021 Flash is primarily used to provide the instruction memory to the Cortex-R4F CPU. It can also be used as static data memory for storing calibration tables. The Flash can be electrically programmed and erased many times to ease code development.

Please refer the following documents for support in how to initialize and use the on-chip Flash and its API:

- *Initialization of Hercules ARM Cortex-R4F Microcontrollers* (literature number )
- *F021 (Texas Instruments 65nm Flash) Flash API Reference Guide* (literature number )

### 5.1.1 Features

- Flash Bank Size: Supports up to 8 Flash banks and 2 Mbyte each
- EEPROM Emulation Bank Size: Supports dedicated EEPROM Emulation Flash banks up to 256 kbytes
- Read, program and erase with a single 3.3v supply voltage
- Supports error detection and correction
  - Single Error Correction and Double Error Detection (SECCDED)
  - ECC (Error Correction Code) is evaluated in the CPU for the main Flash bank arrays and in the Flash Wrapper for the EEPROM emulation Flash banks
  - Address bits included in ECC calculation
- Provides different read modes to optimize performance and verify the integrity of Flash contents
- Provides built-in power mode control logic
- Integrated program/erase state machine
  - Simplifies software algorithms
  - Supports simultaneous read access on a bank while performing a write or erase operation on any one of the remaining banks
  - Suspend command allows read access to a sector being programmed/erased
  - Fast erase and program times (refer to the device datasheet for details)

Please refer to device specific datasheet for the actual size of the Flash memory for the device.

### 5.1.2 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.
- bw - Normal data space bank data width of a Flash bank without error detection check bits. The bw is 128 or 256 bits.
- bwe - EEPROM emulation bank is 64/128 bit wide (without error correction bits).
- Charge pump: Voltage generators and associated control (logic, oscillator, and bandgap, for example).
- CSM: Program/erase command state machine
- Fallback power mode: The power mode (active, standby or sleep, depending on which mode is selected) into which a bank or the charge pump falls back each time the active grace period expires.
- Flash bank: A group of Flash sectors which share input/output buffers, data paths, sense amplifiers, and control logic. Flash bank can store both program instructions and data.
- FEE - Flash EEPROM Emulation. Features on the FMC to support using a Flash type memory in place of an EEPROM Flash memory. EEPROM is erasable by the word while this Flash memory is only erasable by the sector. FEE can be either an FLEE or FLEP type memory. FLEP is the standard bank. The FEE bank is accessible only through Bus 2 in a special address range and always resides in bank 7.
- FLEE or FLEP Flash bank - The standard Flash bank uses the FLEP module as the Flash bank. The FLEE is a dedicated EEPROM emulation bank that is 64/128 bit wide and has only 16 kbyte sectors.

- Flash module: Flash banks, charge pump, and Flash wrapper.
- Flash wrapper: Power and mode control logic, data path, wait logic, and write/erase state machines.
- FMC - Flash Module Controller.
- Command - A sequence of coded instructions to Flash module to execute a certain task.
- FSM (Flash State Machine) - State machine which parses and decodes FSM commands. It executes embedded algorithms and generates control signals to both Flash bank and charge pump during the actual program/erase operation.
- OTP (one-time programmable): A program-only-once Flash sector (cannot be erased)
- PAGP (Pump Active Grace Period): Time (in HCLK cycles) from when the last of the banks have entered fallback power mode until the pump enters a fallback power mode. This can reduce power consumption by the Flash; however, it can also increase access time.
- Pipeline mode: The mode in which Flash is read 64 bits (+ 8 bit ECC) / 128 bits (+ 16 bit ECC) / 256 bits (+32 bit ECC) (depending on the bank width) at a time, providing higher throughput.
- Sector: A contiguous region of Flash memory which must be erased simultaneously.
- Wide\_Word - the width of the data output from the Flash bank. This can be 128, 144, 256 or 288 bits wide for main Flash banks and additionally 64,72,128 or 144 bits wide for FEE bank.
- Standard read mode: The mode assumed when the pipeline mode is disabled. Physically, 64 (+ 8 bit ECC) / 128 (+ 16 bit ECC) / 256 bits of data (+ 32 bit ECC) (depending on the bank width) is read at a time. However, only 32 bits of data is used while the other bits of data are discarded.
- Read Margin 1 mode: More stringent read mode designed for early detection of marginally erased bits.
- Read Margin 0 mode: More stringent read mode designed for early detection of marginally programmed bits.

### 5.1.3 F021 Flash Tools

Texas Instruments provides the following tools for F021 Flash:

- *nowECC*<sup>™</sup> Generation Tool - to generate the Flash ECC from the Flash data.
- *nowFlash*<sup>™</sup> Programming Tool - to erase/program/verify the device Flash content through JTAG.
- Code Composer Studio - the development environment with integrated Flash programming capabilities.
- F021 Flash API Library - a set of software peripheral functions to program/erase the Flash module. Please refer to *F021 Flash API Reference Guide* for more information.

## 5.2 Default Flash Configuration

At power up, the Flash module state exhibits the following properties:

- Wait states are set to 1
- Pipeline mode is disabled
- The Flash content is protected from modification
- Power modes are set to *Active* (no power savings)
- The boot code must initialize the wait states (including data wait states and address wait states) and the desired pipeline mode by initializing the FRDCNTL register to achieve the optimum system performance. This needs to be done before switching to the final device operating frequency. Please refer to *Initialization of Hercules TMS570 Microcontrollers* for more information.

## 5.3 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 cannot be erased. They are typically blank in new parts. The TI OTP sectors are used to contain manufacturing information. They may be read by the customer but can not be programmed or erased. The TI OTP sectors contain settings used by the Flash API to setup the Flash state machine for erase and program operations.

All of these OTP regions are memory mapped to facilitate ease of access by the CPU. They are memory mapped to an offset starting at F000 0000h in the CPU's memory map. The RWAIT value is used to define the number of wait states. Bank 7 starting at offset F020 0000h is dedicated for EEPROM Emulation and to access this region EWAIT has to be configured.

Usually, the Flash memory is located from 00 to 00FF FFFFh, total 16 Mbytes.

Please refer to the device specific datasheet for the Flash Bank/Sectoring information.

## 5.4 Control Registers

This section details the Flash module registers, summarized in [Table 5-1](#). A detailed description of each register and its bits is also provided.

The Flash module control registers can only be read and/or written by the CPU while in privileged mode. Each register begins on a word boundary. All registers are 32-bit, 16-bit and 8-bit accessible. The start address of the Flash module is FFF8 7000h.

**Table 5-1. Flash Control Register Summary**

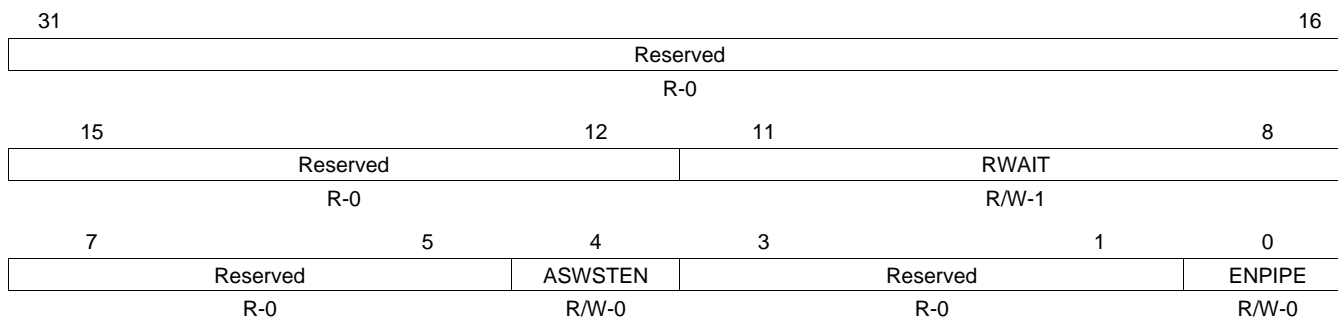
Offset	Acronym	Register Description	Section
FFF8 7000h	FRDCNTL	Flash Option Control Register	<a href="#">Section 5.4.1</a>
FFF8 7008h	FEDACTRL1	Flash Error Detection and Correction Control Register 1	<a href="#">Section 5.4.2</a>
FFF8 700Ch	FEDACTRL2	Flash Error Detection and Correction Control Register 2	<a href="#">Section 5.4.3</a>
FFF8 7020h	FUNC_ERR_ADD	Flash Un-correctable Error Address	<a href="#">Section 5.4.4</a>
FFF8 7030h	FBPROT	Flash Bank Protection Register	<a href="#">Section 5.4.5</a>
FFF8 7034h	FBSE	Flash Bank Sector Enable Register	<a href="#">Section 5.4.6</a>
FFF8 703Ch	FBAC	Flash Bank Access Control Register	<a href="#">Section 5.4.7</a>
FFF8 7040h	FBFALLBACK	Flash Bank Fallback Power Register	<a href="#">Section 5.4.8</a>
FFF8 7048h	FPAC1	Flash Pump Access Control Register 1	<a href="#">Section 5.4.9</a>
FFF8 704Ch	FPAC2	Flash Pump Access Control Register 2	<a href="#">Section 5.4.10</a>
FFF8 7050h	FMAC	Flash Module Access Control Register	<a href="#">Section 5.4.11</a>
FFF8 7054h	FMSTAT	Flash Module Status Register	<a href="#">Section 5.4.12</a>
FFF8 7288h	FSM_WR_ENA	FSM Register Write Enable	<a href="#">Section 5.4.13</a>
FFF8 72B8h	EEPROM_CONFIG	EEPROM Emulation Configuration Register	<a href="#">Section 5.4.14</a>



### 5.4.1 Flash Option Control Register (FRDCNTL)

FRDCNTL supports pipeline mode. There is only one FRDCNTL register for the entire F021 Flash.

**Figure 5-1. Flash Option Control Register (FRDCNTL) [offset = FFF8 7000h]**



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

**Table 5-2. Flash Option Control Register (FRDCNTL) Field Descriptions**

Bit	Field	Value	Description
31-12	Reserved	0	Read returns 0. Writes have no effect.
11-8	RWAIT	0 - Fh	Random/data Read Wait State The random read wait state bits indicate how many wait states are added to a flash read access. In Pipeline mode there is always one wait state even when they are set to 0. <b>Note:</b> The required wait states for each HCLK frequency can be found in device datasheet.
7-5	Reserved	0	Read returns 0. Writes have no effect.
4	ASWSTEN	0 1	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. <b>Note:</b> The required address wait state for each HCLK frequency can be found in device datasheet.
3-1	Reserved	0	Read returns 0. Writes have no effect.
0	ENPIPE	0 1	Enable Pipeline Mode 0 Pipeline mode is disabled. 1 Pipeline mode is enabled.

### 5.4.2 Flash Error Detection and Correction Control Register 1 (FEDACCTRL1)

**Figure 5-2. 32-bit, 4 RowsFlash Error Detection and Correction Control Register 1 (FEDACCTRL1) [offset = FFF8 7008h]**

31	Reserved										25	SUSP_IGNR		24	
R-0											R/W-0				
23	Reserved										EDACMODE		EPEN		16
R-0											R/W-Ah				
15	Reserved										10	EZFEN		9	8
R-0											R/W-0		R/W-0		
7	Reserved		6	EOCV		5	EZCV		EDACEN		0				
R-0		R/W-0		R/W-0		R/W-5h									

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

This register controls ECC parity detection.

**Table 5-3. Flash Error Detection and Correction Control Register 1 (FEDACCTRL1) Field Descriptions**

Bit	Field	Value	Description
31-25	Reserved	0	Read returns 0. Writes have no effect.
24	SUSP_IGNR	0	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. 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	0	Read returns 0. Writes have no effect.
19-16	EDACMODE	5h	Error Correction Mode. During all AXI slave access, flash wrapper will calculate the ECC value from the flash data and send it to the CPU. This is useful while programming the flash and the flash bank contains incomplete words with invalid ECC bytes. <b>Note:</b> These bits have no impact on non AXI slave access.
		All Other Values	The flash wrapper will send the ECC bits in the flash bank to the CPU directly after stripping out the address component. When this field is "5h" the integrity of the flash wrapper is slightly lowered because during AXI slave access, an error in a stored bit is not reflected in the ECC sent to the CPU.
15-10	Reserved	0	Read returns 0. Writes have no effect.
9	EZFEN	0	Correctable errors interrupt enable Single error interrupt is disabled.
		1	Single error interrupt is enabled.
8	EPEN	0	Error Profiling Enable. Error profiling is disabled.
		1	Error profiling is enabled. The correctable error interrupt is generated when number of CPU accesses of correctable bit errors detected and corrected has reached the threshold value defined in the FEDACCTRL2 register.

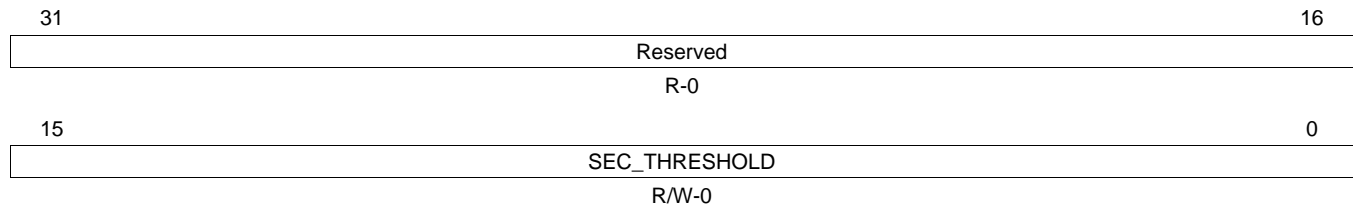
**Table 5-3. Flash Error Detection and Correction Control Register 1 (FEDACCTRL1) Field Descriptions  
(continued)**

Bit	Field	Value	Description
7-6	Reserved	0	Read returns 0. Writes have no effect.
5	EOCV	0	One Condition Valid. 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. During AXI slave accesses, 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. <b>Note:</b> This bit only applies to AXI slave accesses. This bit has no impact during normal CPU ATCM reads.
4	EZCV	0	Zero Condition Valid. 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. During AXI slave accesses, 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. <b>Note:</b> This bit only applies to AXI slave accesses. This bit has no impact during normal CPU ATCM reads.
3-0	EDACEN	5h	Error Detection and Correction Enable CPU single and double error signals are blocked. <b>Note:</b> It is NOT recommended to use mode with ECC in CPU enabled. If the ECC in CPU is enabled, the CPU will still check and correct ECC errors, which would cause reading the wrong data and generating abort.
		13h	Error Detection and Correction is enabled, ECC errors generated from the OTP memory regions and disabled sectors are NOT blocked.
		All Other Values	Error Detection and Correction is enabled, ECC errors generated from the OTP memory regions and disabled sectors are blocked. <b>Note:</b> It is recommended writing "1010" to enable EDACEN to guard against soft errors from flipping EDACEN to a disable state.

This register controls ECC detection.

### 5.4.3 Flash Error Correction and Correction Control Register 2 (FEDACCTRL2)

**Figure 5-3. Flash Error Correction and Correction Control Register 2 (FEDACCTRL2) [offset = FFF8700Ch]**



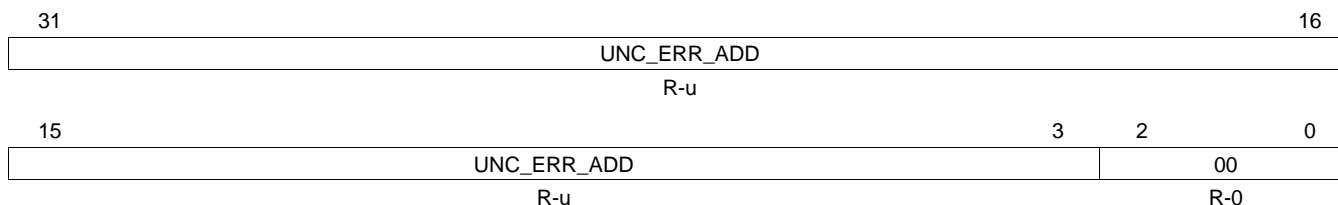
LEGEND: R = Read only; R/W = Read/Write; -n = value after reset

**Table 5-4. Flash Error Correction Control and Correction Register 2 (FEDACCTRL2) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. 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.

#### 5.4.4 Flash Un-correctable Error Address (FUNC\_ERR\_ADD)

**Figure 5-4. Flash Un-correctable Error Address (FUNC\_ERR\_ADD) [offset = FFF8 7020h]**



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

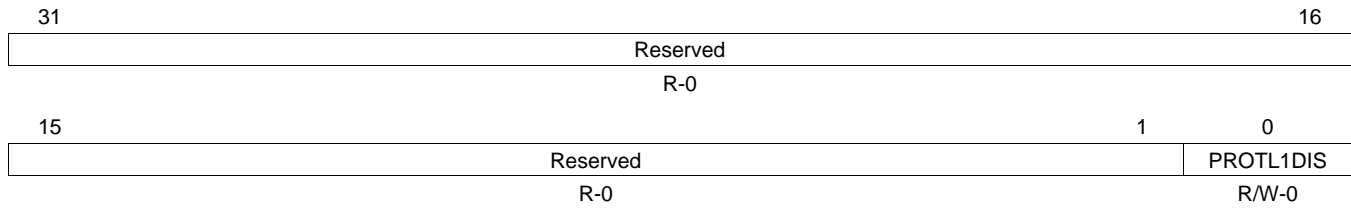
This register is not changed with the reset signal and contains unknown data at powerup.

**Table 5-5. Flash Un-correctable Error Address (FUNC\_ERR\_ADD) Field Descriptions**

Bit	Field	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 in the CPU. The UNC_ERR_ADD also captures the error address when a address bus parity mismatch is detected. This error address is frozen from begin updated until it is read by the CPU. Additional error are blocked until this register is read.  This register captures the full 32 bit incoming address when there is a bus parity error. It only captures address of 22:3 for multiple bit ECC errors. Address parity errors take priority over other errors that happen in the same cycle.
2-0	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.

### 5.4.5 Flash Bank Protection Register (FBPROT)

**Figure 5-5. Flash Bank Protection Register (FBPROT) [offset = FFF8 7030h]**



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

**Table 5-6. Flash Bank Protection Register (FBPROT) Field Descriptions**

Bit	Field	Value	Description
31-1	Reserved	0	Read returns 0. Writes have no effect.
0	PROTL1DIS		PROTL1DIS: Level 1 Protection Disabled  Level 1 Protection Disable bit. Setting this bit disables protection from writing to the OTPPROTDIS bits as well as the Sector Enable registers FBSE for all banks. Clearing this bit enables protection and disables write access to the OTPPROTDIS register bits and FBSE register.
		0	Level 1 protection is disabled.
		1	Level 1 protection is enabled.

### 5.4.6 Flash Bank Sector Enable Register (FBSE)

FBSE provides one enable bit per sector for up to 16 sectors per bank. Each bank in the Flash module has one FBSE register. The bank is selected via the BANK[2:0] bits of the FMAC register. As only one bank at a time can be selected by FMAC, only the register for the bank selected appears at this address.

**Figure 5-6. Flash Bank Sector Enable Register (FBSE) [offset = FFF8 7034h]**

31	Reserved	16
	R-0	
15	BSE	0
	R/W-0	

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

**Table 5-7. Flash Bank Sector Enable Register (FBSE) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	BSE		Bank Sector Enable Each bit corresponds to a flash sector in the bank specified by the FMAC register. 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.

### 5.4.7 Flash Bank Access Control Register (FBAC)

**Figure 5-7. Flash Bank Access Control Register (FBAC) [offset = FFF8 703Ch]**

31	24	23	16
Reserved		OTPPROTDIS	
R-0		R/W-0	
15	8	7	0
BAGP		VREADST	
R/W-0		R/W-Fh	

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

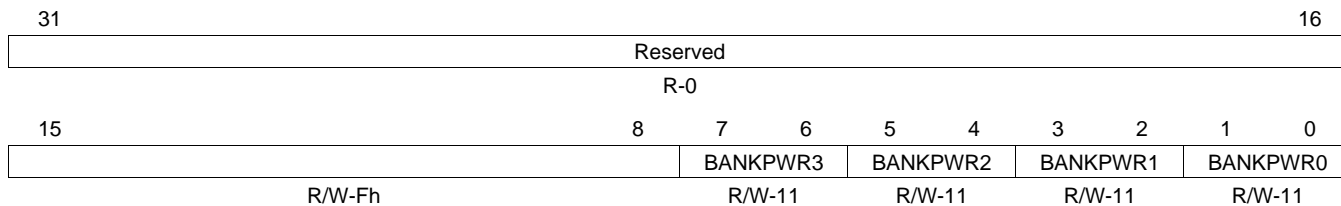
**Table 5-8. Flash Bank Access Control Register (FBAC) Field Descriptions**

Bit	Field	Value	Description
31-24	Reserved	0	Read returns 0. Writes have no effect.
23-16	OTPPROTDIS	0 1	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		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. <b>Note:</b> The prescaled clock used for the BAGP down counter is a clock divided by 16 from HCLK.
7-0	VREADST		VREAD Setup. VREAD is generated by the Flash pump and used for Flash read operation. The bank power up sequencing starts VREADST HCLK cycles after VREAD power supply becomes stable. <b>Note:</b> There is not a programmable Bank Sleep counter and Standby counter register. The number of clock cycles to transition from sleep to standby and standby to active is hardcoded in the Flash wrapper design.



### 5.4.8 Flash Bank Fallback Power Register (FBFALLBACK)

Figure 5-8. Flash Bank Fallback Power Register (FBFALLBACK) [offset = FFF8 7040h]

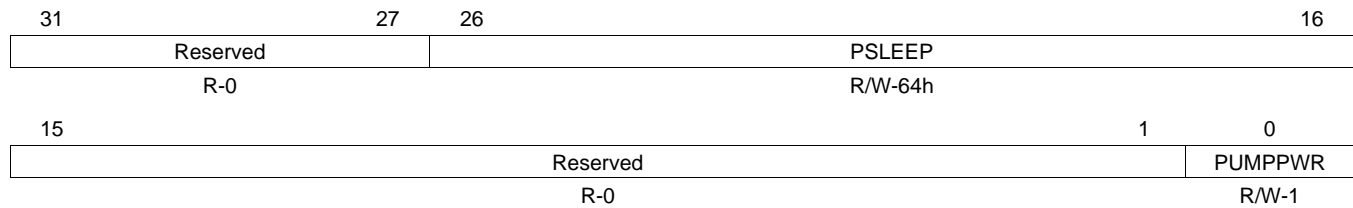


LEGEND: R = Read only; -n = value after reset

Table 5-9. Flash Bank Fallback Power Register (FBFALLBACK) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-6	Reserved		Not used in this device.
	BANKPWR3		Bank 3 Fallback Power Mode
		0	Bank sleep mode
		1h	Bank standby mode
		2h	Reserved
		3h	Bank active mode
5-4	BANKPWR2		Bank 2 Fallback Power Mode - See BANKPWR3[1:0] for details.
3-2	BANKPWR1		Bank 1 Fallback Power Mode - See BANKPWR3[1:0] for details.
1-0	BANKPWR0		Bank 0 Fallback Power Mode - See BANKPWR3[1:0] for details.

### 5.4.9 Flash Pump Access Control Register 1 (FPAC1)

**Figure 5-9. Flash Pump Access Control Register 1 (FPAC1) [offset = FFF8 7048h]**


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

**Table 5-10. Flash Pump Access Control Register 1 (FPAC1) Field Descriptions**

Bit	Field	Value	Description
31-12	Reserved	0	Read returns 0. Writes have no effect.
26-16	PSLEEP		Pump Sleep.  These bits contain the starting count value for the charge pump sleep down counter. While the charge pump is in sleep mode, the power mode management logic holds the charge pump sleep counter at this value. When the charge pump exits sleep power mode, the down counter delays from 0 to PSLEEP pump sleep down clock cycles before putting the charge pump into active power mode.  <b>Note:</b> Pump sleep down counter clock is a divide by 2 input of HCLK. That is, there are 2*HCLK cycles for every PSLEEP counter cycle.
15-1	Reserved	0	Read returns 0. Writes have no effect.
0	PUMPPWR	0	Flash Charge Pump Fallback Power Mode Sleep (all pump circuits disabled)
		1	Active (all pump circuits active)

### 5.4.10 Flash Pump Access Control Register 2 (FPAC2)

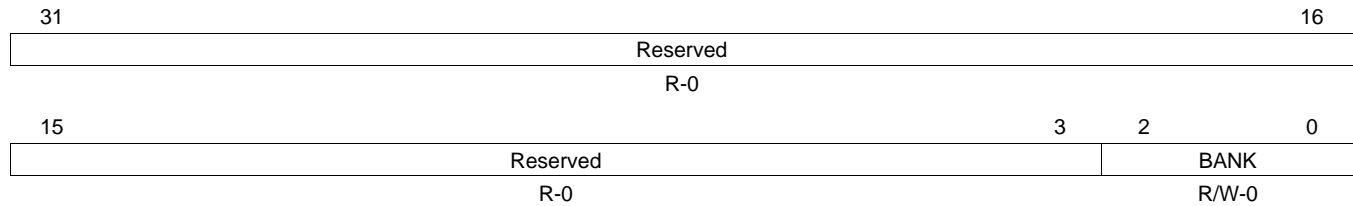
**Figure 5-10. Flash Pump Access Control Register 2 (FPAC2) [offset = FFF8 704Ch]**

31	Reserved	16
	R-0	
15	PAGP	0
	R/W-0	

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

**Table 5-11. Flash Pump Access Control Register 2 (FPAC2) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	PAGP		Pump Active Grace Period This register contains the starting count value for the PAGP mode down counter. Any access to flash memory causes the counter to reload with the PAGP value. After the last access to flash memory, the down counter delays from 0 to 65535 prescaled HCLK clock cycles before entering one of the charge pump fallback power modes as determined by PUMPPWR in the FPAC1 register. <b>Note:</b> The PAGP down counter is clocked by the same prescaled clock as the BAGP down counter which is a divide by 16 of HCLK.

**5.4.11 Flash Module Access Control Register (FMAC)**
**Figure 5-11. Flash Module Access Control Register (FMAC) [offset = FFF8 7050h]**


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

**Table 5-12. Flash Module Access Control Register (FMAC) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
2-0	BANK		Bank Enable. These bits select which bank is enabled for operations such as local register access, OTP sector access, and program/erase commands. These bits select only one bank at a time from up to eight banks depending on the specific device being used. For example, a "000" selects bank 0; "011" selects Bank 3. <b>Note:</b> 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.

### 5.4.12 Flash Module Status Register (FMSTAT)

**Figure 5-12. Flash Module Status Register (FMSTAT) [offset = FFF8 7054h]**

Reserved								
R-0								
31							24	
Reserved								
R-0								
23						18	17	16
Reserved						RVSUSP	RDVER	
R-0						R-0	R-0	
15	14	13	12	11	10	9	8	
RVF	ILA	DBT	PGV	PCV	EV	CV	BUSY	
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	
7	6	5	4	3	2	1	0	
ERS	PGM	INV-DAT	CSTAT	VOLTSTAT	ESUSP	PSUSP	SLOCK	
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	

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

**Table 5-13. Flash Module Status Register (FMSTAT) Field Descriptions**

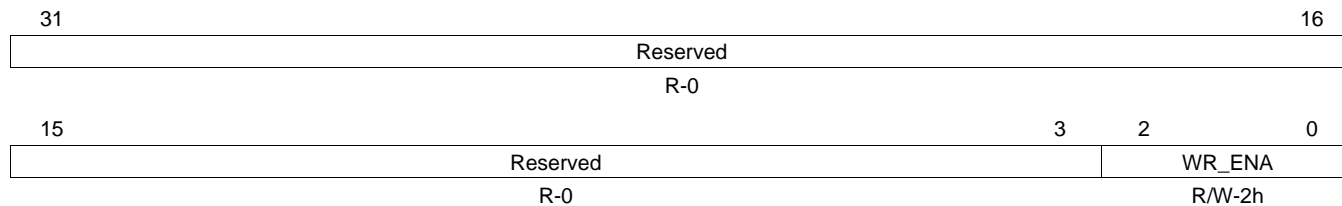
Bit	Field	Value	Description
31-18	Reserved	0	Read returns 0. Writes have no effect.
17	RVSUSP		Read Verify Suspend When set, this bit indicates that the flash module has received and processed a suspend command during a read-verify operation. This bit remains set until the read-verify-resume command has been issued or the Clear_More command is run.
16	RVDER		Read verify command currently underway When set, this bit indicates that the flash module is actively performing a read-verify operation. This bit is set when read-verify starts and is cleared when it is complete. It is also cleared when the read-verify is suspended and set when the read-verify resumes.
15	RVF		Read Verify Failure When set, indicates that a read verify mismatch is detected using the Read Verify command. This bit remains set until clear_status or clear_more FSM commands are run.
14	ILA		Illegal Address When set, indicates that an illegal address is detected. Three conditions can set illegal address flag. <ol style="list-style-type: none"> <li>Writing to a hole (un-implemented logical address space) within a flash bank.</li> <li>Writing to an address location to an un-implemented flash space.</li> <li>Input address for write is decoded to select a different bank from the bank ID register.</li> <li>The address range does not match the type of FSM command. For example, the erase_sector and erase_OTP commands must match the address regions.</li> <li>TI-OTP address selected but CMD_EN in FSM_ST_MACHINE is not set.</li> </ol>
13	DBT		Disturbance Test Fail This bit sets during a Program Sector command when the FSM first reads an address and it is not all ones.
12	PGV		Program Verify When set, indicates that a word is not successfully programmed after the maximum allowed number of program pulses are given for program operation.
11	PCV		Precondition Verify. When set, indicates that a sector is not successfully preconditioned (pre-erased) after the maximum allowed number of program pulses are given for precondition operation for any applied command such as Erase Sector command. During Precondition verify command, this flag is set immediately if a flash bit is found to be 1.
10	EV		Erase Verify When set, indicates that a sector is not successfully erased after the maximum allowed number of erase pulses are given for erase operation. During Erase verify command, this flag is set immediately if a bit is found to be 0.

**Table 5-13. Flash Module Status Register (FMSTAT) Field Descriptions (continued)**

Bit	Field	Value	Description
9	CV		Compact Verify When set, indicates that a sector contains one or more bits in depletion after an erase operation with CMPV_ALLOWED set. During compact verify command, this flag is set immediately if a bit is found to be 1.
8	BUSY		When set, this bit indicates that a program, erase, or suspend operation is being processed.
7	ERS		Erase Active When set, this bit indicates that the flash module is actively performing an erase operation. This bit is set when erasing starts and is cleared when erasing is complete. It is also cleared when the erase is suspended and set when the erase resumes.
6	PGM		Program Active When set, this bit indicates that the flash module is currently performing a program operation. This bit is set when programming starts and is cleared when programming is complete. It is also cleared when programming is suspended and set when programming is resumes.
5	INVDAT		Invalid Data When set, this bit indicates that the user attempted to program a "1" where a "0" was already present. This bit is cleared by the Clear Status command.
4	CSTAT		Command Status Once the FSM starts any failure will set this bit. When set, this bit informs the host that the program, erase, or validate sector command failed and the command was stopped. This bit is cleared by the Clear Status command. For some errors, this will be the only indication of an FSM error because the cause does not fall within the other error bit types.
3	VOLTSTAT		Core Voltage Status When set, this bit indicates that the core voltage generator of the pump power supply dipped below the lower limit allowable during a program or erase operation. This bit is cleared by the Clear Status command.
2	ESUSP		Erase Suspended When set, this bit indicates that the flash module has received and processed an erase suspend operation. This bit remains set until the erase resume command has been issued or until the Clear_More command is run.
1	PSUSP		Program Suspended When set, this bit indicates that the flash module has received and processed a program suspend operation. This bit remains set until the program resume command has been issued or until the Clear_More command is run.
0	SLOCK		Sector Lock Status When set, this bit indicates that the operation was halted because the target sector was locked for erasing and programming either by the sector protect bit or by OTP write protection disable bits. (Bits BSE in FBSE register or OTPPROTDIS in register FBAC). This bit is cleared by the Clear Status command. No SLOCK FSM error will occur if all sectors in a bank erase operation are set to 1. All the sectors will be checked but no SLOCK will be set if no operation occurs due to the SECT_ERASED bits being set to all ones. A SLOCK error will occur if attempting to do a sector erase with either BSE is cleared or SECT_ERASED is set

### 5.4.13 FSM Register Write Enable (FSM\_WR\_ENA)

**Figure 5-13. FSM Register Write Enable (FSM\_WR\_ENA) [offset = FFF8 7288h]**

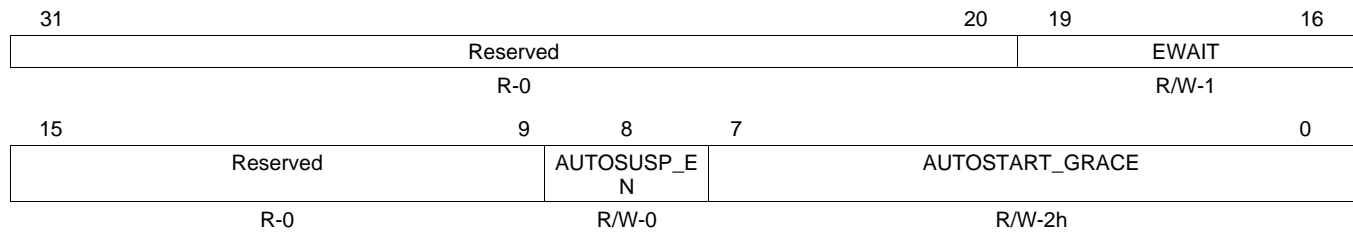


LEGEND: R = Read only; R/W = Read/Write; -n = value after reset

**Table 5-14. Flash Module Access Control Register (FMAC) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
2-0	WR_ENA	2h	FSM Write Enable This register must contain "101" in order to write to any other FSM register. This is the first register to be written when setting up the FSM.
		All other values	For all other values, the FSM registers cannot be written.

### 5.4.14 EEPROM Emulation Configuration Register (EEPROM\_CONFIG)

**Figure 5-14. EEPROM Emulation Configuration Register (EEPROM\_CONFIG) [offset = FFF8 72B8h]**


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

**Table 5-15. EPROM Emulation Configuration Register (EEPROM\_CONFIG) Field Descriptions**

Bit	Field	Value	Description
31-20	Reserved	0	Read returns 0. Writes have no effect.
19-16	EWAIT		EEPROM Wait state Counter This register will replace the RWAIT count in the EEPROM register. The same formulas that apply to RWAIT will apply to EWAIT in the EEPROM bank.
15-9	Reserved	0	Read returns 0. Writes have no effect.
8	AUTOSUSP_EN	0 1	Auto Suspend Enable 0 Auto Suspend Disabled 1 Auto Suspend Enabled. The auto-suspend will begin when the CPU or Bus2 attempts to access a bank with an active and suspendable FSM operation. If this happens the FSM will automatically be issued a suspend command and exit from the FSM. It will then do the access. After the access, the FMC will wait for a time determined by the Autostart_grace field before issuing the FSM resume command.
7-0	AUTOSTART_GRACE	1	Auto-suspend Startup Grace Period The value in this register determines how many cycles the FMC will wait after the last CPU or Bus2 access before issuing the FSM resume command. The FMC will wait 16 HCLK periods for each count in the AUTOSTART_GRACE field. A value of 2 will wait for 32 periods after the last access. Each access will reset the counter to the AUTOSTART_GRACE value * 16.





## ***Tightly-Coupled RAM (TCRAM) Module***

This chapter describes the tightly-coupled RAM (TCRAM) module.

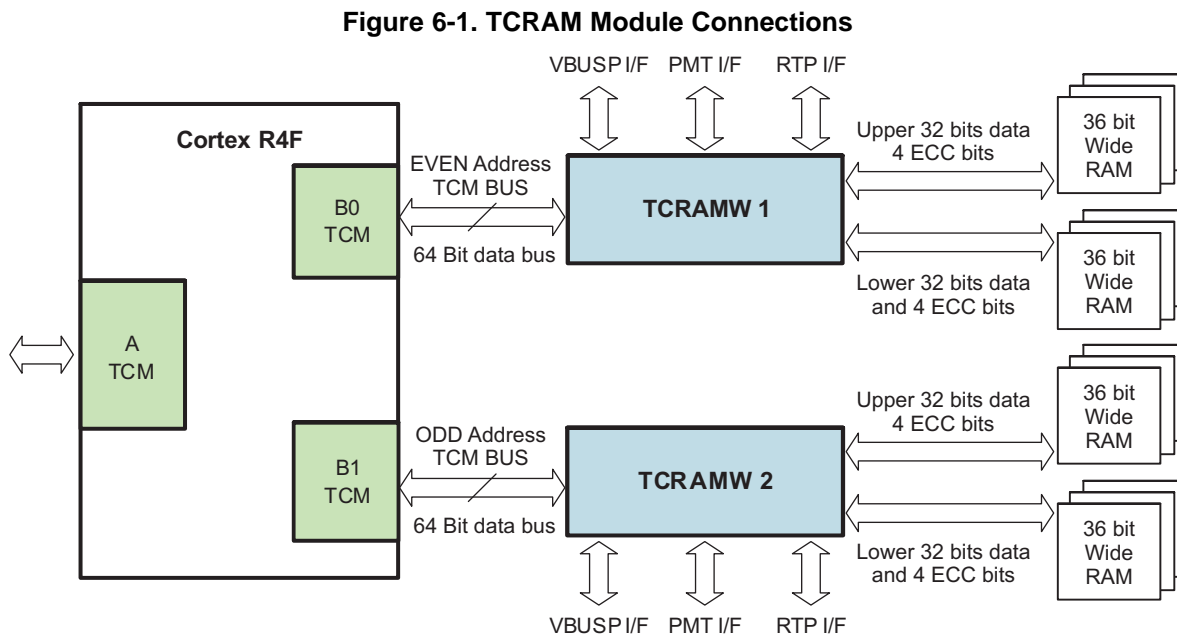
Topic	Page
<b>6.1 Overview .....</b>	<b>296</b>
<b>6.2 RAM Memory Map .....</b>	<b>298</b>
<b>6.3 Safety Features .....</b>	<b>299</b>
<b>6.4 TCRAM Auto-Initialization .....</b>	<b>301</b>
<b>6.5 Trace Module Support .....</b>	<b>301</b>
<b>6.6 Emulation / Debug Mode Behavior .....</b>	<b>301</b>
<b>6.7 Control and Status Registers .....</b>	<b>302</b>

## 6.1 Overview

The Hercules™ family of microcontrollers are based on the ARM® Cortex-R4F™ processor. This CPU has two tightly-coupled memory interfaces – ATCM and BTCM, which are used to interface to the program and data memories, respectively. The Hercules MCUs use the ATCM interface for the main flash memory and the BTCM interface for the CPU data RAM.

### 6.1.1 B0TCM and B1TCM Connection Diagram

The BTCM interface is further divided into two parts – B0TCM and B1TCM, which are both used to interface to actual RAM banks as shown in [Figure 6-1](#).



### 6.1.2 Main Features

The main features of the tightly-coupled RAM interface module are:

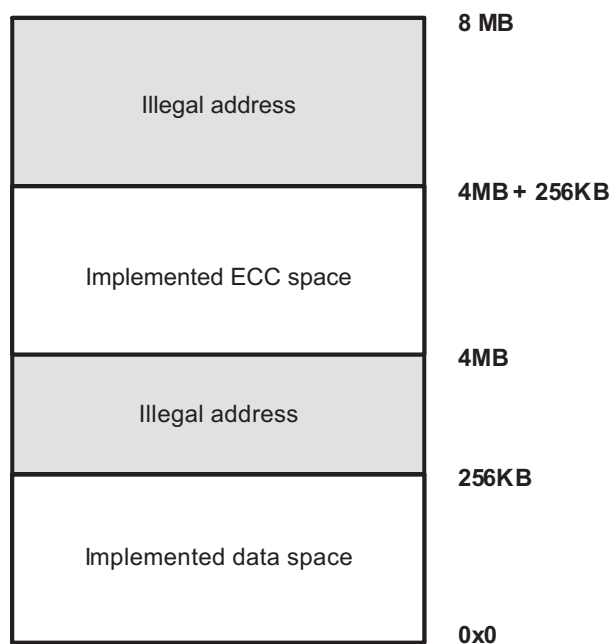
- Controls read/write accesses to the data RAM
- Decodes addresses within the memory region allocated for the RAM
- Supports read and write accesses in 64-bit, 32-bit, 16-bit or 8-bit access sizes
  - Does not support bit-wise operations
- Safety Features:
  - Single-Error-Correction Double-Error-Detection (SECCDED)
    - Uses the CPU's Event bus and maintains the SECCDED status in memory-mapped registers
    - Captures the number of occurrences of single-bit or multi-bit errors as well as the RAM address that has the fault
    - Generates signals for indicating single-bit and multi-bit errors to the Error Signaling Module (ESM)
  - Parity Protection for BTCM Address Bus and Control Signals
    - Uses the CPU's TCM Address Parity Scheme and indicates an address bus parity error to the ESM
  - Redundant Address Decode Scheme
    - Checks the decoding of CPU address lines and generation of correct memory selects for the RAM banks
- Supports auto-initialization of the CPU data RAM banks

- Supports the RAM Trace Port (RTP) Interface
  - Traces out all RAM read and write accesses to the RTP module

## 6.2 RAM Memory Map

The ARM Cortex-R4F CPU allows up to 8MB to be accessed through the BTCM interface. The Hercules family of microcontrollers support up to 256KB RAM on the BTCM interface. Check the specific part's datasheet to identify the actual amount of TCRAM supported on the part. This RAM is protected by ECC allowing the CPU to correct any single-bit errors and detect multi-bit errors within a 64-bit value. The error correction codes (ECC) are stored in the RAM memory space as well. The memory map for the TCRAM and the corresponding ECC space is shown in [Figure 6-2](#). Any access to an unimplemented TCRAM location results in an error response from the TCRAM module.

**Figure 6-2. RAM Memory Map**



Each RAM data word is 64 bits wide. These 64 bits are divided into 32 bits per RAM bank as shown in [TCRAM Module Connections](#). The 8 bits of ECC are also divided into 4 bits per RAM bank.

For every 64-bit read from the RAM, an 8-bit ECC is also read by the CPU on its ECC bus. Similarly, for every 64-bit write to the RAM, the CPU also writes an 8-bit ECC using the same ECC bus.

---

**NOTE: Read-Modify-Write Requirement for Writes to RAM:** The TCRAM interface module supports 64-bit, 32-bit, 16-bit or 8-bit writes to the RAM. However the ECC is calculated by the CPU for 64-bit values only. For any write access smaller than 64 bits, it is necessary to force the CPU to perform a 64-bit read-modify-write operation in order to ensure that the correct ECC is also written. This can be done by setting the bit 1: BTCMRMW of c15, the Secondary Auxiliary Control Register of the CPU.

---

The ECC memory can also be directly accessed via memory-mapped offset addresses starting from 4MB, as shown in . A read from the ECC space results in the 8-bit ECC value appearing on each byte of the 64-bit CPU data. The ECC memory can only be written to as a 64-bit access. The write to the ECC space must also first be enabled via the RAM Control Register (RAMCTRL).

Accesses to the ECC space are not traced out to the RAM Trace Port (RTP).

---

**NOTE: No ECC Error Generated for Accesses to ECC Memory:** A read from the ECC space send the ECC value on both the 64-bit TCM read data bus as well as the 8-bit ECC bus. This could result in the detection of a multi-bit error by the SECDED logic inside the CPU. The TCRAM interface module ignores the ECC error indicated by the CPU for the access to the ECC space.

---

## 6.3 Safety Features

The TCRAM interface module incorporates some features that are designed specifically with safety considerations. These are described in the following sections.

### 6.3.1 Support for Cortex-R4F CPU's Single-Error-Correction Double-Error-Detection (SEDED)

The TCRAM interface module monitors the CPU's event bus. The CPU's event bus signals single-bit or multi-bit errors for B0TCM as well as B1TCM separately. These signals are monitored by the TCRAM modules for each of these interfaces.

#### TCRAM Interface Module Features dedicated for SEDED Support:

- Dedicated single-bit error counter
  - This counter is stored in a memory-mapped register called [RAMOCCUR](#))
  - RAMOCCUR is used to count the single-bit errors corrected by the CPU's SEDED logic
  - The TCRAM interface module allows the application to generate an interrupt via the [RAMINTCTRL](#) register when the number of single-bit errors corrected by the CPU exceeds a programmable threshold, [RAMTHRESHOLD](#)
- RAM Error Status Register
  - The errors detected by the TCRAM interface module as well as those indicated by the CPU are flagged in the [RAMERRSTATUS](#) register
  - There are separate bits to indicate single-bit error, double-bit error, address decode failure, address compare logic failure, read-address parity failure, and write-address parity failure
- ECC Error Address Capture
  - Separate registers to hold the address on which a single-bit error is detected ( [RAMSERRADDR](#)) or a double-bit error is detected ( [RAMUERRADDR](#))
  - The RAMSERRADDR register is only updated when the RAMTHRESHOLD value is set to 1
  - Both the RAMSERRADDR and RAMUERRADDR capture the 64-bit address for the access to the TCRAM as an offset from the base address of the TCRAM (0x08000000 by default)

---

**NOTE: Cortex-R4F CPU Event Bus Signaling Not Enabled By Default:** Upon power-up and after a CPU reset, the event signaling mechanism inside the Cortex-R4F CPU is disabled. This feature must be enabled by setting the Export (X-bit) of the Performance Monitoring and Control Register (PMNC) in the CPU. The TCRAM interface module can only capture the single-bit or double-bit ECC error occurrences once the CPU's event signaling mechanism is enabled.

---

### 6.3.2 Support for Cortex-R4F CPU's Address and Control Bus Parity Checking

The Cortex-R4F CPU calculates a single parity-bit for the TCRAM address and control signals. The TCRAM interface module also computes this parity bit based on the CPU's address bus and control signals. The computed parity bit is compared against the parity bit received from the CPU. A mismatch is signaled as an Address Parity Failure to the Error Signaling Module (ESM). There is a separate address parity failure error channel for B0TCM and B1TCM.

The 64-bit TCRAM address which fails the parity check is captured in the [RAMPERRADDR](#) register as an offset from the base address of the TCRAM (0x08000000 by default). The TCRAM interface module also indicates the type of access, read or write, that failed the parity check. This is indicated by the RADDR PAR FAIL or the WADDR PAR FAIL status flags in the [RAMERRSTATUS](#) register.

The RAMERRSTATUS and RAMPERRADDR registers must be cleared by the application in order for the TCRAM interface module to continue capturing subsequent errors and error addresses.

The parity scheme used for the described parity checking mechanism is defined by the global system parity selection. This can be configured using the DEVPARSEL field of the DEVCR1 control register in the system module. This device-wide parity scheme can be overridden inside the TCRAM interface module by configuring the Address Parity Override field in the [RAMCTRL](#) register.

---

**NOTE: No Change Of Parity Scheme On-The-Fly:** The TCRAM interface module does not support on-the-fly change to the parity scheme being used for checking the CPU address bus and control bus. The application must ensure that the parity polarity (odd or even) is not changed while there is an ongoing access to the TCRAM.

---

### 6.3.3 Redundant Address Decode

The TCRAM interface module generates the memory selects for each of the TCRAM banks as well as the ECC memory based on the CPU address. The logic to generate these memory selects is duplicated and the outputs compared to detect any address decode errors. A mismatch is indicated as an Address Error to the Error Signaling Module (ESM), one signal for B0TCM and one for B1TCM. The TCRAM or ECC address that caused the fault is captured in the [RAMUERRADDR](#) register. This is a 64-bit address is stored as an offset from the base of the TCRAM or ECC memory.

As described earlier, each individual physical RAM bank is 36 bits wide. Each RAM bank contributes 32 bits of data and 4 bits of ECC when the bus master performs a 64-bit read from the TCRAM. Each TCRAM bank receives a memory select and the address from the TCRAM interface module. Any difference between the address and the memory selects results in wrong data and ECC pair being sent to the CPU. The CPU's SECDED block will detect this data error.

The TCRAM interface module also supports a mechanism to test the operation of the redundant address decode logic and the compare logic. This testing is supported by providing a test stimulus, and can be triggered by the application by configuring the [RAMTEST](#) register. The address of any error identified during testing of the redundant address decode and compare logic is not captured in the [RAMUERRADDR](#) register.

---

**NOTE: Address decode checking when in compare logic test mode:** When the address decode and compare logic test mode is enabled, the redundant address decode and compare logic is not available for checking the proper generation of the memory selects for the TCRAM and ECC memory.

---

## 6.4 TCRAM Auto-Initialization

The RAM memory can be initialized by using the dedicated auto-initialization hardware. The TCRAM Module initializes the entire memory when the auto-init is enabled for the RAM. All RAM data memory is initialized to zeros and the ECC memory is initialized to the correct ECC value for zeros, that is, FCh.

## 6.5 Trace Module Support

The TCRAM Module traces out the following signals to the RAM Trace Port (RTP) module, thereby providing RAM dataport trace capability.

- 18-bit address line
- 64-bit data bus
- Byte strobe information
- Current access master identification number
- Access type: Opcode or data fetch
- Read or Write access

No data is traced for an access to ECC memory.

## 6.6 Emulation / Debug Mode Behavior

The following describes the behavior of the TCRAM Module when in debug mode:

- The [RAMOCCUR](#) register continues to count the single-bit error corrections performed by the Cortex-R4F CPU's SECDED logic.
- No single-bit error interrupt is generated nor is any single-bit error address captured even when the RAMOCCUR counter reaches the programmed single-bit error correction threshold.
- No uncorrectable error interrupt is generated nor is any double-bit error address captured.
- No address parity error interrupt is generated nor is any parity error address captured.
- The [RAMUERRADDR](#) register is not cleared by a read in debug mode.
  - That is, if a double-bit error address is captured and is not read by the CPU before entering debug mode, then it remains frozen during debug mode even if it is read.
- The [RAMPERRADDR](#) register is not cleared by a read in debug mode.



## 6.7 Control and Status Registers

The TCRAM Module registers are accessed through the system module registers' space in the Cortex-R4F CPU's memory map. All registers are 32-bit wide and are located on a 32-bit boundary. Reads and writes to registers are supported in 8-, 16-, and 32-bit accesses. The base address for the control registers is FFFF F800h and FFFF F900h for even and off RAM ECC.

**Table 6-1. TCRAM Module Control and Status Register Map**

Offset	Acronym	Register Description	Section
00	RAMCTRL	TCRAM Module Control Register	<a href="#">Section 6.7.1</a>
04h	RAMTHRESHOLD	TCRAM Module Single-Bit Error Correction Threshold Register	<a href="#">Section 6.7.2</a>
08h	RAMOCCUR	TCRAM Module Single-Bit Error Occurrences Control Register	<a href="#">Section 6.7.3</a>
0Ch	RAMINTCTRL	TCRAM Module Interrupt Control Register	<a href="#">Section 6.7.4</a>
10h	RAMERRSTATUS	TCRAM Module Error Status Register	<a href="#">Section 6.7.5</a>
14h	RAMSERRADDR	TCRAM Module Single-Bit Error Address Register	<a href="#">Section 6.7.6</a>
1Ch	RAMUERRADDR	TCRAM Module Uncorrectable Error Address Register	<a href="#">Section 6.7.7</a>
30h	RAMTEST	TCRAM Module Test Mode Control Register	<a href="#">Section 6.7.8</a>
38h	RAMADDRDECVECT	TCRAM Module Test Mode Vector Register	<a href="#">Section 6.7.9</a>
3Ch	RAMPERRADDR	TCRAM Module Parity Error Address Register	<a href="#">Section 6.7.10</a>

### 6.7.1 TCRAM Module Control Register (RAMCTRL)

The RAMCTRL register, shown in Figure 6-3 and described in Table 6-2, controls the safety features supported by the TCRAM Module.

**Figure 6-3. TCRAM Module Control Register (RAMCTRL) [offset = 00]**

31	30	29	28	27	24
Reserved	EMU TRACE DIS	Reserved	ADDR PARITY OVERRIDE		
R-0	R/W-0	R-0	R/W-0		
23	Reserved		20	19	16
R-0			ADDR PARITY DISABLE		
R-0			R/W-5h		
15	Reserved			9	8
R-0				ECC WR EN	
R-0				R/W-0	
7	Reserved		4	3	0
R-0			ECC DETECT EN		
R-0			R/W-Ah		

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

**Table 6-2. TCRAM Module Control Register (RAMCTRL) Field Descriptions**

Bit	Field	Value	Description
31	Reserved	0	Read returns 0. Writes have no effect.
30	EMU TRACE DIS	0 1	Emulation Mode Trace Disable. This bit, when set, disables the tracing of read data to RAM Trace Port (RTP) module for an emulation mode access. Data is allowed to be traced out to the trace modules for emulation mode accesses. Data is blocked from being traced out to the trace modules for emulation mode accesses.
29-28	Reserved	0	Read returns 0. Writes have no effect.
27-24	ADDR PARITY OVERRIDE	Dh All Others	Address Parity Override. This field, when set to Dh, will invert the parity scheme selected by the device global parity selection. The address parity checker would then work on the inverted parity scheme. By default, the parity scheme is the same as the global device parity scheme. Parity scheme is opposite to the device global parity scheme. Parity scheme is the same as the device global parity scheme.
23-20	Reserved	0	Read returns 0. Writes have no effect.
19-16	ADDR PARITY DISABLE	Ah All Others	Address Parity Detect Disable. This field, when set to Ah, disables the parity checking for the address bus. The parity checking is enabled when this field is set to any other value. <b>Note: The application must ensure that the parity error address register is cleared before enabling address parity checking.</b> Address parity checking is disabled Address parity checking is enabled
15-9	Reserved	0	Read returns 0. Writes have no effect.
8	ECC WR EN	0 1	ECC Memory Write Enable. This bit is provided to prevent accidental writes to the ECC memory. A write access to the ECC memory is allowed only when the ECC WR EN bit is set to 1. If this bit is cleared, then any writes to ECC memory are ignored. Note: Reads are allowed from the ECC memory regardless of the state of the ECC WR EN. ECC memory writes are disabled. ECC memory writes are enabled.
7-4	Reserved	0	Read returns 0. Writes have no effect.

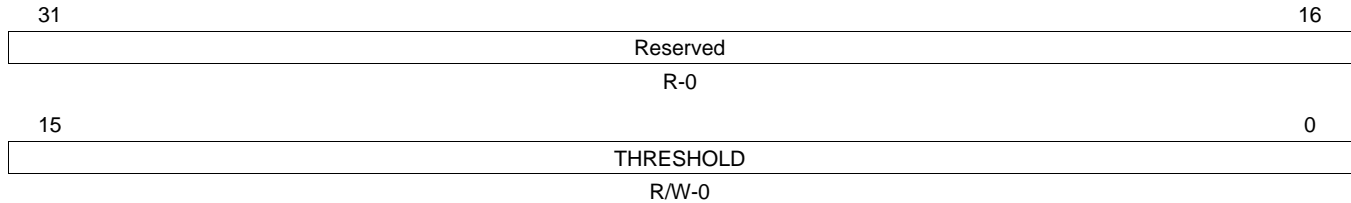
**Table 6-2. TCRAM Module Control Register (RAMCTRL) Field Descriptions (continued)**

Bit	Field	Value	Description
3-0	ECC DETECT EN		ECC Detect Enable. This is a 4-bit key to enable the ECC detection feature in the TCRAM Module. If this field is set to any value other than 5h, then the TCRAM Module starts monitoring the TCM event bus and generates the corresponding error status flags. The error status updates are done only when the ECC DETECT EN field is not 5h. The ECC detection is enabled by default as the ECC DETECT EN default value is Ah.
		5h	ECC detection is disabled.
		All Others	ECC detection is enabled.

### 6.7.2 TCRAM Module Single-Bit Error Correction Threshold Register (RAMTHRESHOLD)

The RAMTHRESHOLD register, shown in [Figure 6-4](#) and described in [Table 6-3](#), allows the application to configure the number of single-bit error corrections by the SECDED logic inside the Cortex-R4F CPU before generating a single-bit error interrupt.

**Figure 6-4. TCRAM Module Single-Bit Error Correction Threshold Register (RAMTHRESHOLD)  
[offset = 04h]**



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

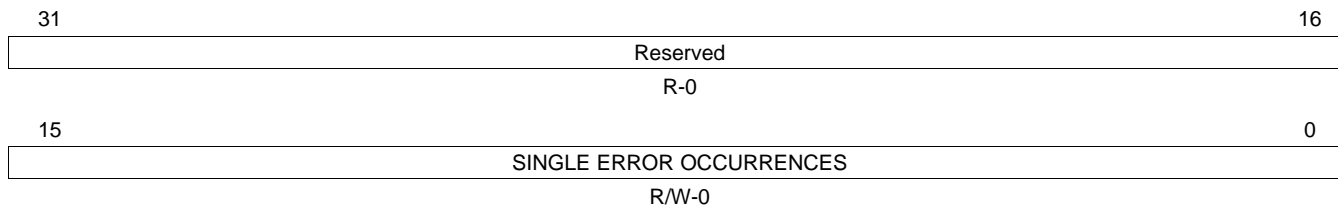
**Table 6-3. TCRAM Module Single-Bit Error Correction Threshold Register (RAMTHRESHOLD) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	THRESHOLD		Single-bit Error Threshold Count. This field contains the threshold value for the Single-bit Error Correction (SEC) occurrences before the single-bit error interrupt is generated. If this threshold is set to 1 then all single-bit error addresses are captured. To enable the error occurrence detection, the threshold must be set to a non-zero value.

### 6.7.3 TCRAM Module Single-Bit Error Occurrences Counter Register (RAMOCCUR)

The RAMOCCUR register, shown in Figure 6-5 and described in Table 6-4, indicates the number of single-bit error corrections performed by the SEC logic inside the Cortex-R4F CPU.

**Figure 6-5. TCRAM Module Single-Bit Error Occurrences Counter Register (RAMOCCUR) [offset = 08h]**



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

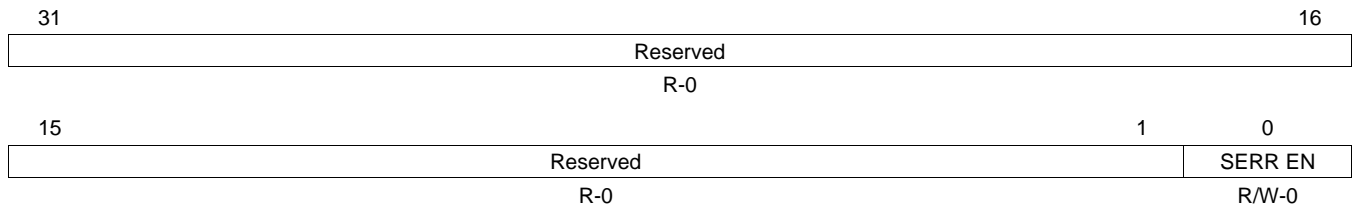
**Table 6-4. TCRAM Module Single-Bit Error Occurrences Counter Register (RAMOCCUR) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	SINGLE ERROR OCCURRENCES		<p>Single-bit Error Correction Occurrences. This 16-bit counter contains the number of single-bit error occurrences. RAMOCCUR is reset to zero when it becomes equal to the THRESHOLD value set in the RAMTHRESHOLD register. The application must clear the RAMOCCUR register by writing 0x0 before setting the THRESHOLD value. If the RAMOCCUR value is already higher than the programmed THRESHOLD value then the counter increments and wraps around (overflow) to zero.</p> <p><b>Note: If the application tries to clear the RAMOCCUR register at the same time as the TCRAM Module tried to update it, then the TCRAM Module takes priority.</b></p> <p><b>Note: When the RAMTHRESHOLD register is set to 1, then the RAMOCCUR register must be cleared whenever a single-bit error correction occurs in order to count subsequent single-bit error corrections.</b></p>

**6.7.4 TCRAM Module Interrupt Control Register (RAMINTCTRL)**

The RAMINTCTRL register, shown in Figure 6-6 and described in Table 6-5, enables the generation of an interrupt to the CPU whenever the number of single-bit error corrections (RAMOCCUR) reaches the programmed threshold (RAMTHRESHOLD).

**Figure 6-6. TCRAM Module Interrupt Control Register (RAMINTCTRL) [offset = 0Ch]**



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

**Table 6-5. TCRAM Module Interrupt Control Register (RAMINTCTRL) Field Descriptions**

Bit	Field	Value	Description
31-15	Reserved	0	Read returns 0. Writes have no effect.
0	SERR EN	0	Single-bit Error Correction Interrupt Enable. This bit, when set to 1, enables the generation of the single-bit error interrupt when the RAMOCCUR count reaches the programmed RAMTHRESHOLD. If the interrupt is not enabled, the single-bit error counter continues to count by resetting back to zero without generating any error interrupt. The SERR status flag in the RAMERRSTATUS register gets set regardless of whether the SERR interrupt is enabled or not.
		1	Single-bit error generation is enabled.

### 6.7.5 TCRAM Module Error Status Register (RAMERRSTATUS)

The RAMERRSTATUS register, shown in Figure 6-7 and described in Table 6-6, indicates the status of the various error conditions monitored by the TCRAM Module.

**Figure 6-7. TCRAM Module Error Status Register (RAMERRSTATUS) [offset = 10h]**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															
Reserved										WADDR PAR FAIL		RADDR PAR FAIL			
R-0										R/W-0		R/W-0			
Reserved		DERR		ADDR COMP LOGIC FAIL		Reserved		ADDR DEC FAIL		Reserved		SERR			
R-0		R/W-0		R/W-0		R-0		R/W-0		R-0		R/W-0			

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

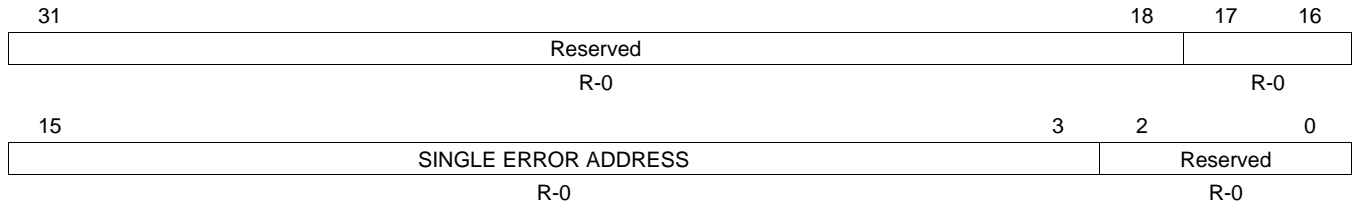
**Table 6-6. TCRAM Module Error Status Register (RAMERRSTATUS) Field Descriptions**

Bit	Field	Value	Description
31-10	Reserved	0	Read returns 0. Writes have no effect.
9	WADDR PAR FAIL		This bit indicates a Write Address Parity Failure. This bit must be cleared by writing 1 to it in order to enable the capture of parity error address for subsequent failures. This bit must be in a cleared state for generation of any new parity error interrupt.
8	RADDR PAR FAIL		This bit indicates a Read Address Parity Failure. This bit must be cleared by writing 1 to it in order to enable the capture of parity error address for subsequent failures. This bit must be in a cleared state for generation of any new parity error interrupt.
7-6	Reserved	0	Read returns 0. Writes have no effect.
5	DERR		This bit indicates a multi-bit error detected by the Cortex-R4F SECCDED logic.
4	ADDR COMP LOGIC FAIL		Address decode logic element failed. This bit indicates that the redundant address decode logic test scheme has detected that a compare element has malfunctioned during the testing of the logic. This bit has to be cleared by writing 1 to it in order to enable the capture of uncorrectable error address for subsequent failures. This bit has to be in a cleared state for generation of a new uncorrectable error interrupt. This bit only gets set in the test mode, and has no relevance in functional mode.
3	Reserved	0	Read returns 0. Writes have no effect.
2	ADDR DEC FAIL		Address decode failed. This bit indicates that an address error interrupt was generated by the redundant address decode and compare logic due to a functional failure. This bit must be cleared by writing 1 to it in order to enable the capture of uncorrectable error address for subsequent failures. This bit has to be in a cleared state for generation of a new address error interrupt.
1	Reserved	0	Read returns 0. Writes have no effect.
0	SERR		Single Error Status. This bit indicates that the single-bit error threshold has been reached. This bit is set even if the single-bit error threshold interrupt is disabled. This bit must be cleared by writing 1 to it in order to clear the interrupt request and to enable subsequent single-bit error interrupt generation.

### 6.7.6 TCRAM Module Single-Bit Error Address Register (RAMSERRADDR)

The RAMSERRADDR register, shown in [Figure 6-8](#) and described in [Table 6-7](#), captures the address for which the Cortex-R4F CPU detected a single-bit error.

**Figure 6-8. TCRAM Module Single-Bit Error Address Register (RAMSERRADDR) [offset = 14h]**



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

**Table 6-7. TCRAM Module Single-Bit Error Address Register (RAMSERRADDR) Field Descriptions**

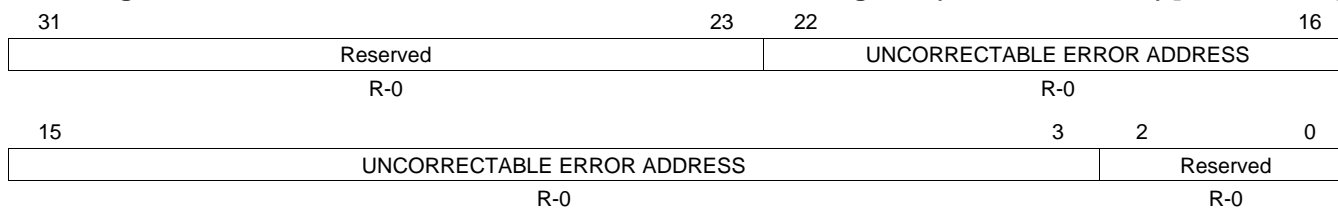
Bit	Field	Value	Description
31-18	Reserved	0	Read returns 0. Writes have no effect.
17-3	SINGLE ERROR ADDRESS		This register captures the bits 17-3 of the address for which the Cortex-R4F CPU detects a single-bit error when the RAMTHRESHOLD register is set to 1. The lower 3 bits are always tied to zero so that the address captured is a double-word (64-bit) address. This is a 64-bit address is stored as an offset from the base of the TCRAM or ECC memory.  This register can only be reset by asserting power-on reset, and holds the last error address even after a system reset.
2-0	Reserved	0	Read returns 0. Writes have no effect.



### 6.7.7 TCRAM Module Uncorrectable Error Address Register (RAMUERRADDR)

The RAMUERRADDR register, shown in [Figure 6-9](#) and described in [Table 6-8](#), captures the address for which the Cortex-R4F CPU detected a multi-bit error.

**Figure 6-9. TCRAM Module Uncorrectable Error Address Register (RAMUERRADDR) [offset = 1Ch]**



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

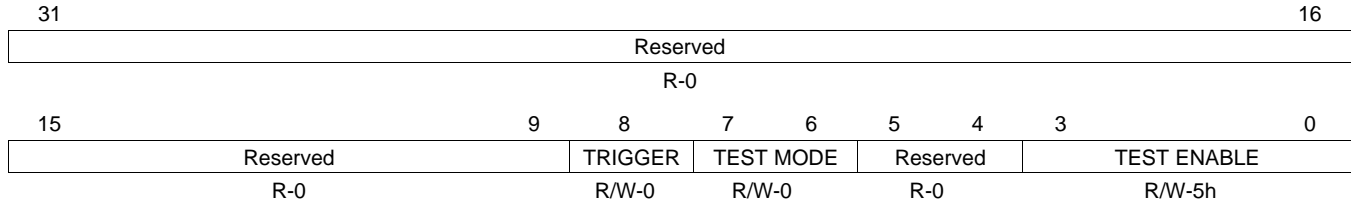
**Table 6-8. TCRAM Module Uncorrectable Error Address Register (RAMUERRADDR) Field Descriptions**

Bit	Field	Value	Description
31-23	Reserved	0	Read returns 0. Writes have no effect.
22-3	UNCORRECTABLE ERROR ADDRESS		<p>This register captures the address for which there was an uncorrectable error or an address error.</p> <p>The uncorrectable error is indicated by the Cortex-R4F CPU's SECDDED logic and the address error is indicated by the TCRAM Module's redundant address decode and compare logic.</p> <p>For the SECDDED multi-bit or double-bit uncorrectable error this register stores the bits 17-3 of the TCM access address. The lower 3 bits 2-0 are always read as zeros to indicate that the latched address is a double-word address. The address bits 31-18 are read as zeros. This is a 64-bit address is stored as an offset from the base of the TCRAM or ECC memory.</p> <p>For a redundant address decode and compare logic error this register stores the complete TCM access address rounded to a double-word boundary (bits 22-3). This error is also indicated by the ADDR DEC FAIL status flag in the RAMERRSTATUS register.</p> <p>The register has to be read-cleared to enable further error address captures. Reading the register does not clear its contents but enables the register to be updated with an uncorrectable error address.</p> <p>This register can only be reset by asserting power-on reset, and holds the last error address even after a system reset.</p>
2-0	Reserved	0	Read returns 0. Writes have no effect.

### 6.7.8 TCRAM Module Test Mode Control Register (RAMTEST)

The RAMTEST register, shown in Figure 6-10 and described in Table 6-9, controls the test mode of the TCRAM Module.

**Figure 6-10. TCRAM Module Test Mode Control Register (RAMTEST) [offset = 30h]**



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

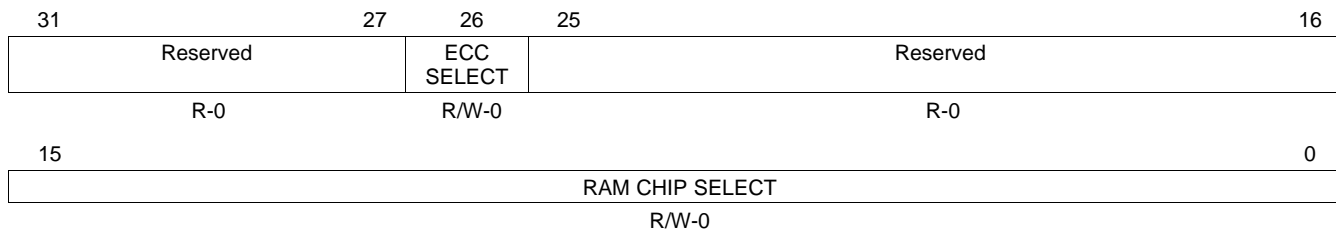
**Table 6-9. TCRAM Module Test Mode Control Register (RAMTEST) Field Descriptions**

Bit	Field	Value	Description
31-9	Reserved	0	Read returns 0. Writes have no effect.
8	TRIGGER		Test Trigger. This is an auto reset test trigger used to test the redundant address decode and compare logic. A redundant address decode test is executed when test mode is enabled and the test trigger is applied by writing a 1 to this bit. The trigger is valid only if test is enabled and the correct mode is configured in the TEST MODE field, and the RAMERRSTATUS and RAMUERRADDR registers are in the cleared state.
7-6	TEST MODE		Test Mode. This field selects either equality of inequality testing schemes.  If TEST MODE is set to 2h, equality check is done. The test stimulus stored in ADDRTEST_VECT register is fed directly to both the channels of the comparator. If the XOR of these two inputs <b>is not zero</b> then UERR interrupt is generated and ADDR COMP LOGIC FAIL flag is set in RAMERRSTATUS register.  If TEST MODE is set to 1h, inequality check is done. The test stimulus stored in ADDRTEST_VECT register is inverted and fed into one channel and the non-inverted vector is fed into the other channel. If the XOR of these inputs <b>is zero</b> then the UERR interrupt is generated and ADDR COMP LOGIC FAIL flag is set in RAMERRSTATUS register.
5-4	Reserved	0	Read returns 0. Writes have no effect.
3-0	TEST ENABLE	5h All Others	Test Enable. This is a 4-bit key to enable the redundant address decode and compare logic test scheme. If the test scheme is enabled then the compare logic uses the test vector inputs from the ADDRTEST_VECT register. The functional path comparison is disabled when test mode is enabled.  Test mode is enabled.  Test mode is disabled.

### 6.7.9 TCRAM Module Test Mode Vector Register (RAMADDRDECVECT)

The RAMADDRDECVECT register, shown in Figure 6-11 and described in Table 6-10, is used for testing the redundant address decode and compare logic of the TCRAM Module.

**Figure 6-11. TCRAM Module Test Mode Vector Register (RAMADDRDECVECT) [offset = 38h]**



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

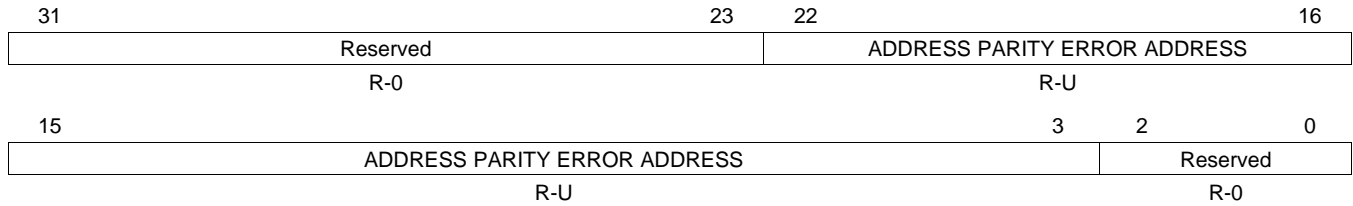
**Table 6-10. TCRAM Module Test Mode Vector Register (RAMADDRDECVECT) Field Descriptions**

Bit	Field	Value	Description
31-27	Reserved	0	Read returns 0. Writes have no effect.
26	ECC SELECT		ECC Select. This bit is used to store the ECC select value for the redundant address decode and compare logic. The stored value is passed as test stimulus for the built-in test scheme.
25-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	RAM CHIP SELECT		RAM Chip Select. This field is used to store the RAM chip select value for the redundant address decode and compare logic. The stored value is passed as test stimulus for the built-in test scheme.

**6.7.10 TCRAM Module Parity Error Address Register (RAMPERRADDR)**

The RAMPERRADDR register, shown in Figure 6-12 and described in Table 6-11, stores the address for which an address-parity error was detected.

**Figure 6-12. TCRAM Module Parity Error Address Register (RAMPERRADDR) [offset = 3Ch]**



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

**Table 6-11. TCRAM Module Parity Error Address Register (RAMPERRADDR) Field Descriptions**

Bit	Field	Value	Description
31-23	Reserved	0	Read returns 0. Writes have no effect.
22-3	ADDRESS PARITY ERROR ADDRESS		Parity Error Address. This register stores the double-word boundary (bits 22-3) of the TCM access address for which there was an address parity error. This register must be read-cleared to enable further error address captures. Reading the register does not clear the register contents but enables the register to be updated with a new parity error address. This is a 64-bit address is stored as an offset from the base of the TCRAM or ECC memory.
2-0	Reserved	0	Read returns 0. Writes have no effect.



## **Programmable Built-In Self-Test (PBIST) Module**

---



---

This chapter describes the programmable built-in self-test (PBIST) controller module used for testing the on-chip memories on the Hercules microcontrollers.

Topic	Page
7.1 Overview .....	316
7.2 RAM Grouping and Algorithm .....	318
7.3 PBIST Flow .....	321
7.4 Memory Test Algorithms on the On-chip ROM .....	324
7.5 PBIST Control Registers .....	326
7.6 PBIST Configuration Example .....	340

## 7.1 Overview

The PBIST (Programmable Built-In Self-Test) controller architecture provides a run-time-programmable memory BIST engine for varying levels of coverage across many embedded memory instances.

### 7.1.1 Features of PBIST

- Information regarding on-chip memories, memory groupings, memory background patterns and test algorithms stored in dedicated on-chip PBIST ROM
- Host processor interface to configure and start BIST of memories
- Supports testing of PBIST ROM itself as well
- Supports testing of each memory at its maximum access speed in application
- Implements intelligent clock gating to conserve power
- Execution of microcode from PBIST ROM supported for ROM clock speeds up to 112MHz

### 7.1.2 PBIST vs. Application Software-Based Testing

The PBIST architecture consists of a small coprocessor with a dedicated instruction set targeted specifically toward testing memories. This coprocessor executes test routines stored in the PBIST ROM and runs them on multiple on-chip memory instances. The on-chip memory configuration information is also stored in the PBIST ROM.

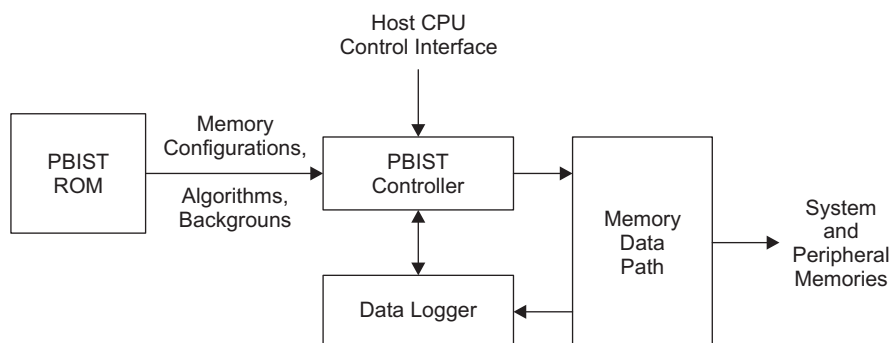
The PBIST Controller architecture offers significant advantages over tests running on the main Cortex-R4F processor (application software-based testing):

- Embedded CPUs have a long access path to memories outside the tightly-couple memory sub-system, while the PBIST controller has a dedicated path to the memories specifically for the self-test
- Embedded CPUs are designed for their targeted use and are often not easily programmed for memory test algorithms.
- The memory test algorithm code on embedded CPUs is typically significantly larger than that needed for PBIST.
- The embedded CPU is significantly larger than the PBIST controller.

### 7.1.3 PBIST Block Diagram

The [Figure 7-1](#) illustrates the basic PBIST blocks and its wrapper logic for the device.

**Figure 7-1. PBIST Block Diagram**



#### 7.1.3.1 The On-chip ROM

The on-chip ROM contains the information regarding the algorithms and memories to be tested.

### 7.1.3.2 Host Processor Interface to the PBIST Controller Registers

The Cortex-R4F CPU can select the algorithm and RAM groups for the memories' self-test from the on-chip ROM based on the application requirements. Once the self-test has executed, the CPU can query the PBIST controller registers to identify any memories that failed the self-test and to then take appropriate next steps as required by the application's author.

### 7.1.3.3 Memory Data Path

This is the read and write data path logic between different system and peripheral memories tightly coupled to the PBIST memory interface. The PBIST controller executes each selected algorithm on each valid memory group sequentially until all the algorithms are executed.

---

**NOTE:** Not all algorithms are designed to run on all RAM groups. If an algorithm is selected to run on an incompatible memory, this will result in a failure. Refer to [Table 7-1](#) and [Table 7-2](#) for RAM grouping and algorithm information.

---



## 7.2 RAM Grouping and Algorithm

Table 7-1 gives the list of RAM groups and their types supported on the device. Table 7-2 maps the different algorithms supported in application mode for the RAM groups with the background patterns used for the particular algorithm.

**Table 7-1. RAM Grouping**

RAM Group	Module	Memory Type	RGS	RDS
1	PBIST ROM	ROM	1	0
2	STC ROM	ROM	2	0
3	DCAN1	Two Port	3	0..5
4	DCAN2	Two Port	4	0..5
5	DCAN3	Two Port	5	0..5
6	ESRAM1	Single Port	6	0
7	MIBSPI1	Two Port	7	0..3
8	MIBSPI3	Two Port	7	4..7
9	MIBSPI5	Two Port	7	8..11
10	VIM	Two Port	8	0..1
11	MibADC1	Two Port	9	0
12	DMA	Two Port	10	0..5
13	N2HET1	Two Port	11	0..11
14	HET TU1	Two Port	12	0..5
15	RTP	Two Port	13	0..8
18	MIBADC2	Two Port	15	0
19	N2HET2	Two Port	16	0..11
20	HET TU2	Two Port	17	0..5
21	ESRAM5	Single Port	18	0
22	ESRAM6	Single Port	19	0
23	ETHERNET	Two Port	20	0..2
24			20	3..5
25		Single Port	20	6
26	USB	Two Port	21	1..2
27		Single Port	21	0
28	ESRAM8	Single Port	25	0

**Table 7-2. Algorithm Mapping**

Algorithm No	Module	Memory Type	Valid RAM Groups	Available background patterns	Valid RAMINFO register value
1	triple_read_slow_read	ROM	1,2		0x00000003
2	triple_read_fast_read	ROM	1,2		0x00000003
3	march13n_red	Two Port	3,4,5,6,7,8,9,10,11,12,13,14,15,16,18,19,20,23,24,26	0x00000000, 0x96699669, 0x0F0F0F0F, 0xAA55AA55, 0xC3C3C3C3	0x02CEFFDC
4	march13n_red	Single Port	6,17,21,22,25,27,28	0x00000000, 0x96699669, 0x0F0F0F0F, 0xAA55AA55, 0xC3C3C3C3	0x0D310020
5	down1A_red	Two Port	3,4,5,6,7,8,9,10,11,12,13,14,15,16,18,19,20,23,24,26	0xFFFFFFFF, 0xAAAAAAAA	0x02CEFFDC

**Table 7-2. Algorithm Mapping (continued)**

Algorithm No	Module	Memory Type	Valid RAM Groups	Available background patterns	Valid RAMINFO register value
6	down1A_red	Single Port	6,17,21,22,25,27,28	0xFFFFFFFF, 0xAAAAAAAA	0x0D310020
7	mapcolumn	Two Port	3,4,5,6,7,8,9,10,11,12,13 ,14,15,16,18,19,20,23, 24,26	0xFFFFFFFF, 0x00000000	0x02CEFFDC
8	mapcolumn	Single Port	6,17,21,22,25,27,28	0xFFFFFFFF, 0x00000000	0x0D310020
9	precharge	Two Port	3,4,5,6,7,8,9,10,11,12,13 ,14,15,16,18,19,20,23, 24,26	0xFFFFFFFF, 0x00000000	0x02CEFFDC
10	precharge	Single Port	6,17,21,22,25,27,28	0xFFFFFFFF, 0x00000000	0x0D310020
11	dtxn2	Two Port	3,4,5,6,7,8,9,10,11,12,13 ,14,15,16,18,19,20,23, 24,26	0xFFFFFFFF, 0x00000000	0x02CEFFDC
12	dtxn2	Single Port	6,17,21,22,25,27,28	0xFFFFFFFF, 0x00000000	0x0D310020
13	pmos_open	Two Port	3,4,5,6,7,8,9,10,11,12,13 ,14,15,16,18,19,20,23, 24,26	0xFFFFFFFF, 0x00000000	0x028AE5DC
14	pmos_open	Single Port	6,17,21,22,25,27,28	0xFFFFFFFF, 0x00000000	0x0D310020
15	pmos_open_slice1	Two Port	10,12,13,19,23	0xFFFFFFFF, 0x00000000	0x00441A00
16	pmos_open_slice2	Two Port	10,12,13,19,23	0xFFFFFFFF, 0x00000000	0x00441A00
17	flip10	Two Port	3,4,5,6,7,8,9,10,11,12,13 ,14,15,16,18,19,20,23, 24,26	0xFFFFFFFF	0x02CEFFDC
18	flip10	Single Port	6,17,21,22,25,27,28	0xFFFFFFFF	0x0D310020
19	iddq	Two Port	3,4,5,6,7,8,9,10,11,12,13 ,14,15,16,18,19,20,23, 24,26	0x00000000	0x02CEFFDC
20	iddq	Single Port	6,17,21,22,25,27,28	0x00000000	0x0D310020
21	retention	Two Port	3,4,5,6,7,8,9,10,11,12,13 ,14,15,16,18,19,20,23, 24,26	0x00000000	0x02CEFFDC
22	retention	Single Port	6,17,21,22,25,27,28	0x00000000	0x0D310020
23	iddq	Two Port	3,4,5,6,7,8,9,10,11,12,13 ,14,15,16,18,19,20,23, 24,26	0xFFFFFFFF	0x02CEFFDC
24	iddq	Single Port	6,17,21,22,25,27,28	0xFFFFFFFF	0x0D310020
25	retention	Two Port	3,4,5,6,7,8,9,10,11,12,13 ,14,15,16,18,19,20,23, 24,26	0xFFFFFFFF	0x02CEFFDC
26	retention	Single Port	6,17,21,22,25,27,28	0xFFFFFFFF	0x0D310020
27	iddqrowstripe	Two Port	3,4,5,6,7,8,9,10,11,12,13 ,14,15,16,18,19,20,23, 24,26	0x00000000	0x02CEFFDC
28	iddqrowstripe	Single Port	6,17,21,22,25,27,28	0x00000000	0x0D310020
29	iddqrowstripe	Two Port	3,4,5,6,7,8,9,10,11,12,13 ,14,15,16,18,19,20,23, 24,26	0xFFFFFFFF	0x02CEFFDC
30	iddqrowstripe	Single Port	6,17,21,22,25,27,28	0xFFFFFFFF	0x0D310020

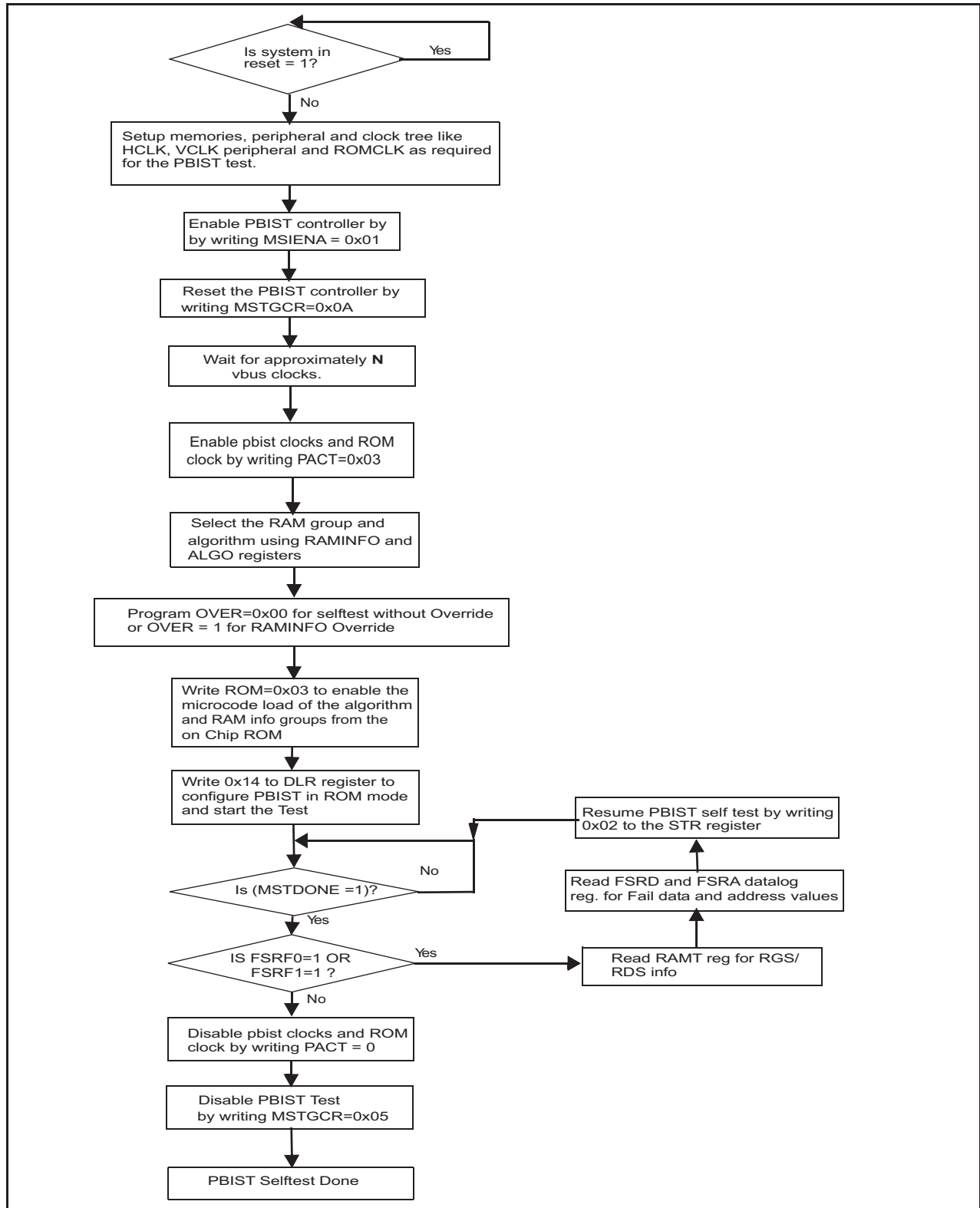
**NOTE:** March13 is the most recommended algorithm for the memory self-test.

---

### 7.3 PBIST Flow

The Figure 7-1 illustrates the memory self-test flow.

Figure 7-2. PBIST Memory Self-Test Flow Diagram



### 7.3.1 PBIST Sequence

1. Configure the device clock sources and domains so that they are running at their target frequencies.
2. Program the HCLK to PBIST ROM clock ratio by configuring the ROM\_DIV field (bits 9:8) of the MSTGCR register of the system module. This device supports a max PBIST ROM clock frequency of 110MHz.
3. Enable PBIST Controller by setting bit 1 of MSIENA register in system module.
4. Enable the PBIST self-test by writing a value of 0x0A to bits 3:0 of the MSTGCR in the system module.
5. Wait for N VBUS clock cycles based on the HCLK to PBIST ROM clock ratio:
  - N = 16 when HCLK:PBIST ROM clock is 1:1
  - N = 32 when HCLK:PBIST ROM clock is 1:2
  - N = 64 when HCLK:PBIST ROM clock is 1:4
  - N = 64 when HCLK:PBIST ROM clock is 1:8
6. Write 3h to PACT register to enable the PBIST internal clocks and ROM interface clock.
7. Program the ALGO register to decide which algorithm from the instruction ROM must be selected (the default value of ALGO register is all 1's, meaning all algorithms are selected). Similarly, program the RINFOL and RINFOU registers to indicate whether a particular RAM group in the instruction ROM would get executed or not.

---

**NOTE:** In case of RAM Override (Override Register (OVER) = 00), the user should make sure that only the algorithms that run on similar RAMs are selected. If a single port algorithm is selected in ROM Algorithm Mask Register (ALGO), the RAM Info Mask Lower Register (RINFOL) and RAM Info Mask Upper Register (RINFOU) must select only the single port RAM's. The same applies for two port RAM's. Check [Table 7-1](#) for information on the memory types.

---

8. Program OVER = 1h to run PBIST self-test without RAM override. Program OVER = 0 to run PBIST self-test with RAM Override.
9. Write a value of 3h to the ROM mask register should the microcode for the Algorithms as well as the RAM groups loaded from the on chip PBIST ROM.
10. Write DLR (Data Logger register) with 14h to configure the PBIST run in ROM mode and to enable the config access. This starts the memory self-tests.
11. Wait for the PBIST self-test done by polling MSTDONE bit of MSTCGSTAT register in System Module.
12. Once self-test is completed, check the Fail Status registers FSRF0 and FSRF1.
  - In case there is a failure (FSRF0 or FSRF1 = 1h)
    - (a) Read RAMT register which indicates the RGS and RDS values of the failure RAM
    - (b) Read FSRC0 and FSRC1 registers which contains the failure count
    - (c) Read FSRA0 and FSRA1 registers which contains the address of first failure
    - (d) Read FSRDL0 and FSRDL1 registers which contains the failure data.
    - (e) Write a value of 2h to the STR register to resume the test.
  - In case there is no failure (FSRF0 and FSRF1 = 0) the memory self-test is completed.
    - (a) Disable the PBIST internal and ROM clocks by writing a 0 to the PACT register.
    - (b) Disable the PBIST self-test by writing a value of 5h to bits 3:0 of the MSTGCR in the system module
- Repeat steps 2 through 9 for subsequent runs with different RAM group and algorithm configurations.
13. After required Memory tests are completed, Resume or Start the Normal Application software.

---

**NOTE:** The contents of the selected memory before the test will be completely lost. User software must take care of data backup if required. Typically the PBIST tests are carried out at the beginning of Application software.

---

---

**NOTE:** Memory test fail information is reported in terms of RGS:RDS and not RAM GROUP. Check [Table 7-1](#) for information on the RGS:RDS information applicable to each memory being tested.

---

## 7.4 Memory Test Algorithms on the On-chip ROM

This section provides a brief description for some of the test algorithms used for memory self-test.

### 1. **March13N:**

- March13N is the baseline test algorithm for SRAM testing. It provides the highest overall coverage. The other algorithms provide additional coverage of otherwise missed boundary conditions of the SRAM operation.
- The concept behind the general march algorithm is to indicate:
  - The bit cell can be written and read as both a 1 and a 0.
  - The bits around the bit cell do not affect the bit cell.
- The basic operation of the march is to initialize the array to a know pattern, then march a different pattern through the memory.
- Type of faults detected by this algorithm:
  - Address decoder faults
  - Stuck-At faults
  - Coupled faults
  - State coupling faults
  - Parametric faults
  - Write recovery faults
  - Read/write logic faults

### 2. **Map Column:**

- The MAP COLUMN algorithm is used to identify bit line sensitivities in the memory array. The memory array is loaded with a row stripe pattern of all 1s in the first row followed by all 0s in the second row and repeated throughout the array. Then the values are read down each column on consecutive cycles. The pattern in memory is inverted and run the column reads again.
- This particular pattern is looking for the following SRAM failure mechanisms:
  - Leakage due to a low resist path in a bit
  - An Open in the bit cell
  - Leakage on a BIT or BITN line
  - Miss-balance in the sense amp
  - Leakage in the sense
  - High resist in the sense amp
  - Failure of the pre-charge circuits after read operations

### 3. **Pre-Charge:**

- The Pre-Charge algorithm exercises the pre-charge capability within the SRAM array. It is important to specifically target this issue as it is the only part of the analog portion of the SRAM that is frequency sensitive.
- Similar to the MAP COLUMN algorithm, this algorithm works its way down the columns of the SRAM. However, unlike the MAP COLUMN, this algorithm sandwiches a write between two reads to force the worst-case conditions for the pre-charge circuits in the array.
- This test will fail when an increase in system frequency nears the minimum access time of the array, at this boundary:
  - High voltage should operate better than low voltage.
  - Likewise, low temperature should operate better than high temperature.
- If devices fail this test within the operational range of the CPU, then this is a very good test for characterizing the CPU/memory system.

### 4. **DOWN1a:**

- The Down1 pattern forces the switching of all data bits and most address bits on consecutive read cycles. This is primarily a read/write test of the CPU/memory subsystem.

- The aggressive writes target at-speed write failures.
- It also targets row/column decode in the memory array.
- Targets the sense amps and sense amp multiplexors.
- Memory array output buffers.
- This algorithm operates as follows:
  - Load 1st half of the memory under test with one pattern.
  - Load 2nd half of the memory under test with the bit-wise inverse of the pattern.
  - Alternate sequential reads sequences between one sequence starting at the first of the array and a second sequence starting at the end of the array.
  - Upon completion of the read back, invert the patterns in both halves of the array and repeat the above step.
  - Perform an aggressive write sequence by alternate write between the bottom half of the memory upwards with an data pattern and the top half of the memory downwards with the inverse data pattern.
  - Invert the data pattern for the above two steps to performing another sequence of aggressive writes.

#### 5. DTXN2a:

This algorithm is used to target the global column decode Logic.



## 7.5 PBIST Control Registers

PBIST controller uses configuration registers for programming the algorithm and its execution. All the configuration registers are memory mapped for access by the CPU through the Peripheral Bus interface. The base address for the control registers is FFFF E400h.

---

**NOTE:** There is no watchdog functionality implemented in the PBIST controller. If a bad code is executed, the PBIST will run forever. The PBIST controller does not guard against this situation.

---



---

**NOTE:** Registers are accessible only when the clock to PBIST controller is active. The clock is activated by first writing 0x3 to PACT register.

---

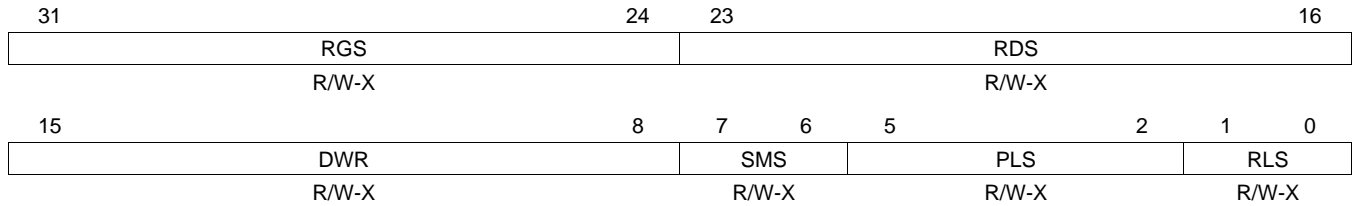
**Table 7-3. PBIST Registers**

Offset	Acronym	Register Description	Section
000h - 15Ch	Reserved	Reserved locations. Do not write to these locations.	
160h	RAMT	RAM Configuration Register	<a href="#">Section 7.5.1</a>
164h	DLR	Datalogger Register	<a href="#">Section 7.5.2</a>
168h - 17Ch	Reserved	Reserved locations. Do not write to these locations.	
180h	PACT	PBIST Activate/ROM Clock Enable Register	<a href="#">Section 7.5.3</a>
184h	PBISTID	PBIST ID Register	<a href="#">Section 7.5.4</a>
188h	OVER	Override Register	<a href="#">Section 7.5.5</a>
190h	FSRF0	Fail Status Fail Register 0	<a href="#">Section 7.5.6</a>
194h	FSRF1	Fail Status Fail Register 1	<a href="#">Section 7.5.6</a>
198h	FSRC0	Fail Status Count Register 0	<a href="#">Section 7.5.7</a>
19Ch	FSRC1	Fail Status Count Register 1	<a href="#">Section 7.5.7</a>
1A0h	FSRA0	Fail Status Address 0 Register	<a href="#">Section 7.5.8</a>
1A4h	FSRA1	Fail Status Address 1 Register	<a href="#">Section 7.5.8</a>
1A8h	FSRDL0	Fail Status Data Register 0	<a href="#">Section 7.5.9</a>
1B0h	FSRDL1	Fail Status Data Register 1	<a href="#">Section 7.5.9</a>
1C0h	ROM	ROM Mask Register	<a href="#">Section 7.5.10</a>
1C4h	ALGO	ROM Algorithm Mask Register	<a href="#">Section 7.5.11</a>
1C8h	RINFOL	RAM Info Mask Lower Register	<a href="#">Section 7.5.12</a>
1CCh	RINFOU	RAM Info Mask Upper Register	<a href="#">Section 7.5.13</a>

### 7.5.1 RAM Configuration Register (RAMT)

This register is divided into following internal registers, none of which have a default value after reset. [Figure 7-3](#) and [Table 7-4](#) illustrate this register.

**Figure 7-3. RAM Configuration Register (RAMT) [offset = 0160h]**



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

**Table 7-4. RAM Configuration Register (RAMT) Field Descriptions**

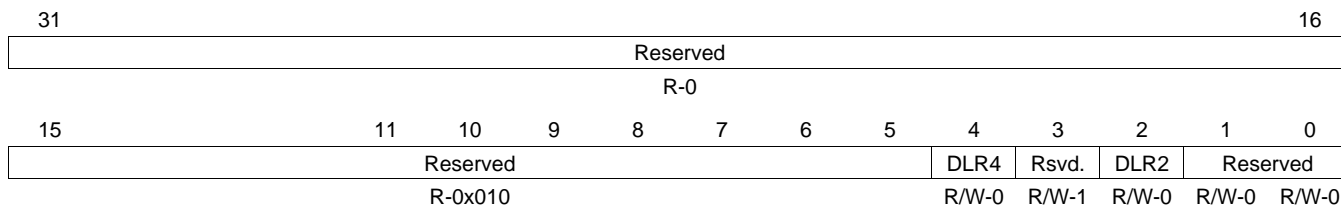
Bit	Field	Value	Description
31-24	RGS		Ram Group Select. Refer <a href="#">Table 7-1</a> for information on the RGS value for each memory.
23-16	RDS		Return Data Select. Refer <a href="#">Table 7-1</a> for information on the RDS values for each memory.
15-8	DWR		Data Width Register
7-6	SMS		Sense Margin Select Register
5-2	PLS		Pipeline Latency Select
1-0	RLS		RAM Latency Select

This register provides the information regarding the memory being currently tested. In case of a PBIST failure, the application can read this register to identify the RGS:RDS values for the memory that failed the self-test.

## 7.5.2 Datalogger Register (DLR)

This register puts the PBIST controller into the appropriate comparison modes for data logging. Figure 7-4 and Table 7-5 illustrate this register.

**Figure 7-4. Datalogger Register (DLR) [offset = 0164h]**



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

**Table 7-5. Datalogger Register (DLR) Field Descriptions**

Bit	Field	Value	Description
31 - 5	Reserved	0	Reads return 0. Do not change these bits from their default value.
4	DLR4		Config access: setting this bit allows the host processor to configure the PBIST controller registers
3	Reserved	1	Do not change this bit from its default value of 1.
2	DLR2		ROM-based testing: setting this bit enables the PBIST controller to execute test algorithms that are stored in the PBIST ROM.
1 - 0	Reserved	00	Do not change these bits from their default value of 00.

- **DLR2: ROM-based testing mode**

Writing a 1 to this register starts the ROM-based testing. This register is used to initiate ROM-based testing from Config and ATE interfaces. Also, since a 1 in this bit position means the instruction ROM is used for memory testing, all the intermediate interrupts and PBIST done signal after each memory test are masked until all the selected algorithms in the ROM are executed for all RAM groups. However, a failure would stop the test and report the status immediately.

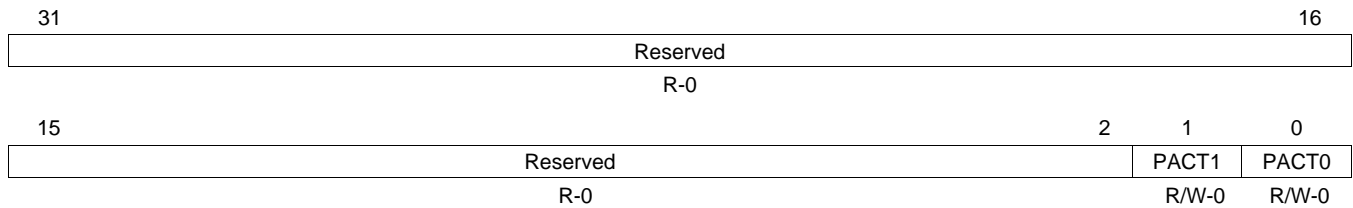
- **DLR4: Config access mode**

This mode, when set, indicates the CPU is being used to access PBIST.

### 7.5.3 PBIST Activate/ROM Clock Enable Register (PACT)

This is the first register that needs to be programmed to activate the PBIST controller. Bit [0] is used for static clock gating, and unless a '1' is written to this bit, all the internal PBIST clocks are shut off. Bit [1] is for turning on the clock going to the instruction ROM. Figure 7-5 and Table 7-6 illustrate this register.

Figure 7-5. PBIST Activate/ROM Clock Enable Register (PACT) [offset = 0180h]



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

Table 7-6. PBIST Activate/ROM Clock Enable Register (PACT) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Read returns 0. Writes have no effect.
1	PACT1	0	PBIST Activate Clock to on chip ROM is disabled
		1	Normal PBIST operation for ROM based testing
0	PACT0	0	ROM Clock Enable Register Disable internal PBIST clocks
		1	Enable internal PBIST clocks

• **PACT0**

This bit must be set to turn on internal PBIST clocks. Setting this bit asserts an internal signal that is used as the clock gate enable. As long as this bit is 0, any access to PBIST will not go through, and PBIST will remain in an almost zero-power mode.

• **PACT1**

Setting this bit turns on the clock going to the instruction ROM.

---

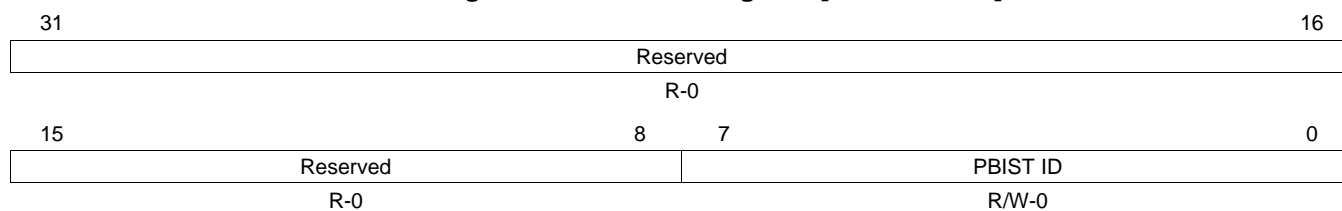
**NOTE:** This register must be programmed to 3h during application self-test.

---

### 7.5.4 PBIST ID Register

Functionality of the register is described in [Figure 7-6](#) and [Table 7-7](#).

**Figure 7-6. PBIST ID Register [offset = 184h]**



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

**Table 7-7. PBIST ID Register Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved		Reads return 0. Writes have no effect.
7-0	PBIST ID		This is a unique ID assigned to each PBIST controller in a device with multiple PBIST controllers.

### 7.5.5 Override Register (OVER)

Functionality of the register is described in [Figure 7-7](#) and [Table 7-8](#).

**Figure 7-7. Override Register (OVER) [offset = 0188h]**

31	Reserved				16		
R-0							
15	Reserved			3	2	1	0
R-0							
OVER0							
			R/W-0	R/W-0	R/W-1		

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

**Table 7-8. Override Register (OVER) Field Descriptions**

Bit	Field	Value	Description
31-3	Reserved	0	Read returns 0. Writes have no effect.
2	Reserved	0	Reserved. This bit must not be changed from its default value of 0.
1	Reserved	0	Reserved. This bit must not be changed from its default value of 0.
0	OVER0	0	RINFO Override Bit The RAM info registers RINFOL and RINFOU are used to select the memories for test
		1	The memory information available from ROM will override the RAM selection from the RAM info registers RINFOL and RINFOU.

• **OVER0**

While doing ROM-based testing, each algorithm downloaded from the ROM has a memory mask associated with it that defines the applicable memory groups the algorithm will be run on. By default, this bit is set to 1, which means the memory mask that is downloaded from the ROM will overwrite the RAM info registers. The override bit can be reset by writing a 0 to it. In this case, the application can select the RAM groups to be tested by configuring the RAM info registers.

**NOTE:** When this override bit = 0, each algorithm selected in ALGO register will run on each RAM selected in RINFOL and RINFOU register. It must be ensured that:

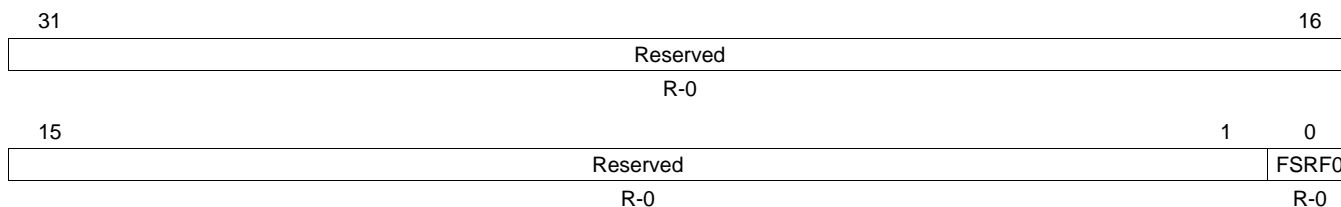
1. Only the same type of memories (single port or two port) are selected, and
2. Only memories that are valid for all algorithms enabled via the ALGO register are selected.

If the above two requirements are not met, the memory self-test will fail.

### 7.5.6 Fail Status Fail Registers (FSRF0 and FSRF1)

These registers indicate if a failure occurred during a memory self-test. Bit 0 gets set whenever a failure occurs. FSRF0 indicates Port0 failures and FSRF1 indicates Port1 failures. The PBIST controller has these two ports in order to optimize testing of single-port and dual-port RAMs. Figure 7-8 and Table 7-9 illustrate the FSRF0 register, while Figure 7-9 and Table 7-10 illustrate the FSRF1 register.

**Figure 7-8. Fail Status Fail Register 0 (FSRF0) [offset = 0190h]**

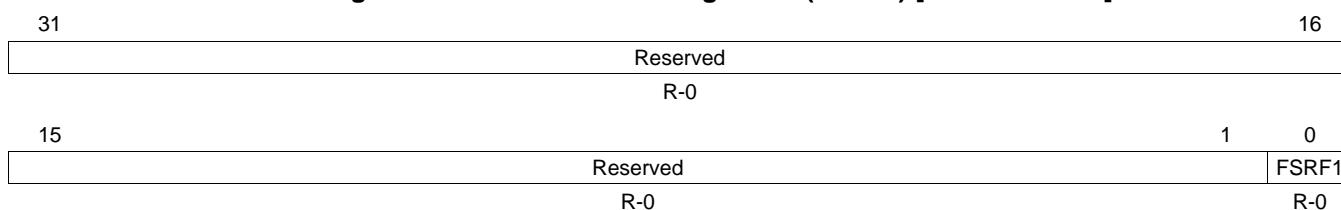


LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-9. Fail Status Fail Register 0 (FSRF0) Field Descriptions**

Bit	Field	Value	Description
31-1	Reserved	0	Read returns 0. Writes have no effect.
0	FSRF0	0	Fail Status 0. This bit would be cleared by reset of the module using MSTGCR register in system module.
		0	No Failure Occured
		1	Indicates a failure on Port 0

**Figure 7-9. Fail Status Fail Register 1 (FSRF1) [offset = 0194h]**



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

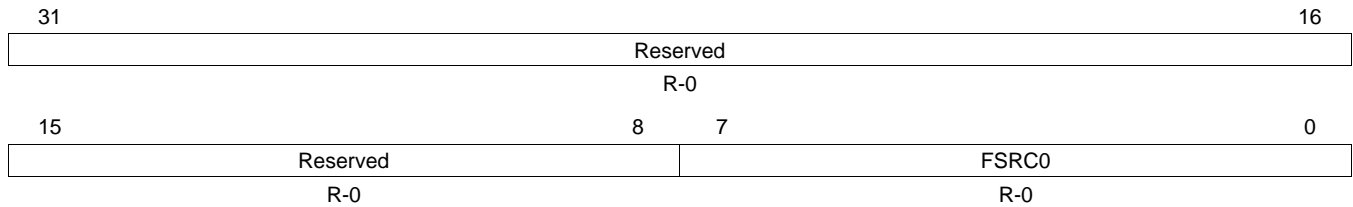
**Table 7-10. Fail Status Fail Register 1 (FSRF1) Field Descriptions**

Bit	Field	Value	Description
31-1	Reserved	0	Read returns 0. Writes have no effect.
0	FSRF1	0	Fail Status 1. This bit would be cleared by reset of the module using MSTGCR register in system module.
		0	No Failure Occured
		1	Indicates a failure on Port 1

### 7.5.7 Fail Status Count Registers (FSRC0 and FSRC1)

These registers keep count of the number of failures observed during the memory self-test. The PBIST controller stops executing the memory self-test whenever a failure occurs in any memory instance for any of the test algorithms. The value in FSRC0 / FSRC1 gets incremented by one whenever a failure occurs and gets decremented by one when the failure is processed. FSRC0 is for Port 0 and FSRC1 is for Port 1. Figure 7-10 and Table 7-11 illustrate the FSRC0 register, while Figure 7-11 and Table 7-12 illustrate the FSRC1 register.

**Figure 7-10. Fail Status Count 0 Register (FSRC0) [offset = 0198h]**

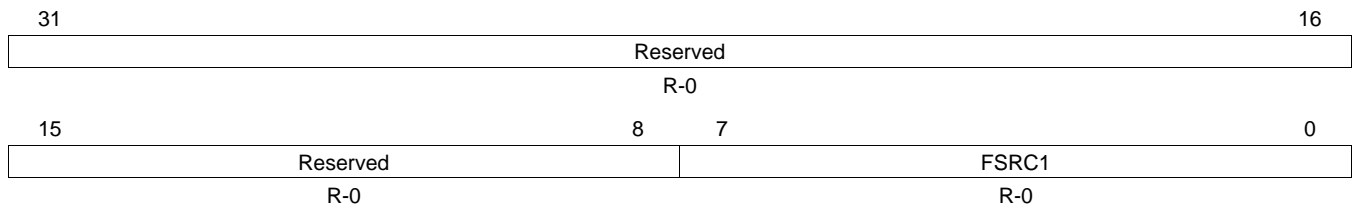


LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-11. Fail Status Count 0 Register (FSRC0) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved		Reads return 0. Writes have no effect.
7-0	FSRC0		Fail Status Count 0. Indicates the number of failures on port 0.

**Figure 7-11. Fail Status Count Register 1 (FSRC1) [offset = 019Ch]**



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

**Table 7-12. Fail Status Count Register 1 (FSRC1) Field Descriptions**

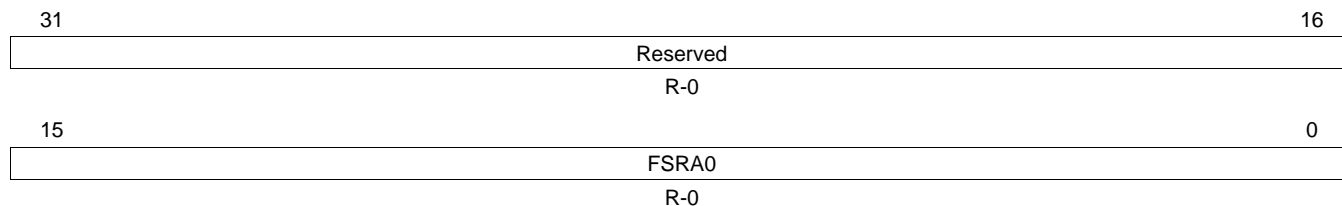
Bit	Field	Value	Description
31-8	Reserved		Reads return 0. Writes have no effect.
7-0	FSRC1		Fail Status Count 1. Indicates the number of failures on port 1.



### 7.5.8 Fail Status Address Registers (FSRA0 and FSRA1)

These registers capture the memory address of the first failure on port 0 and port 1 respectively. Figure 7-12 and Table 7-13 illustrate the FSRA0 register, while Figure 7-13 and Table 7-14 illustrate the FSRA1 register.

**Figure 7-12. Fail Status Address 0 Register (FSRA0) [offset = 01A0h]**

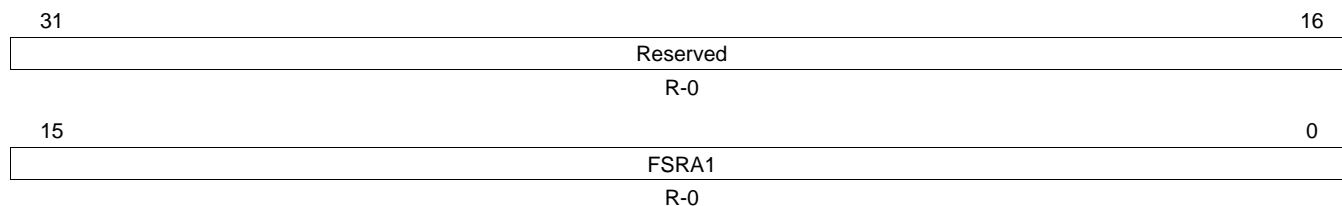


LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-13. Fail Status Address 0 Register (FSRA0) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	FSRA0		Fail Status Address 0. Contains the address of the first failure.

**Figure 7-13. Fail Status Address 1 Register (FSRA1) [offset = 01A4h]**



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

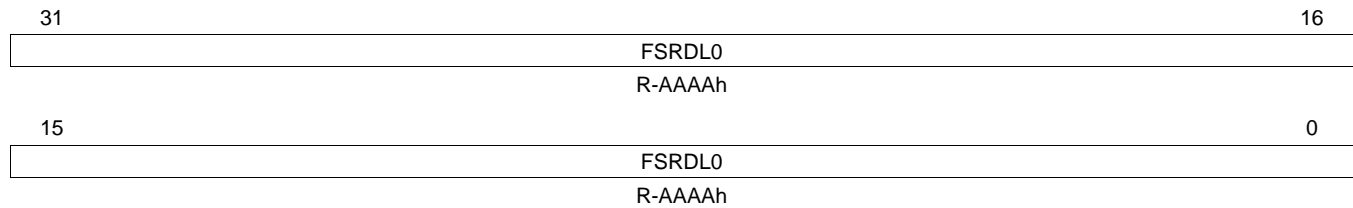
**Table 7-14. Fail Status Address 1 Register (FSRA1) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	FSRA1		Fail Status Address 1. Contains the address of the first failure.

### 7.5.9 Fail Status Data Registers (FSRDLO and FSRDL1)

These registers are used to capture the failure data in case of a memory self-test failure. FSRDLO corresponds to Port 0, while FSRDL1 corresponds to Port 1. [Figure 7-14](#) and [Table 7-15](#) illustrate the FSRDLO register, while [Figure 7-15](#) and [Table 7-16](#) illustrate the FSRDL1 register.

**Figure 7-14. Fail Status Data Register 0 (FSRDLO) [offset = 01A8h]**

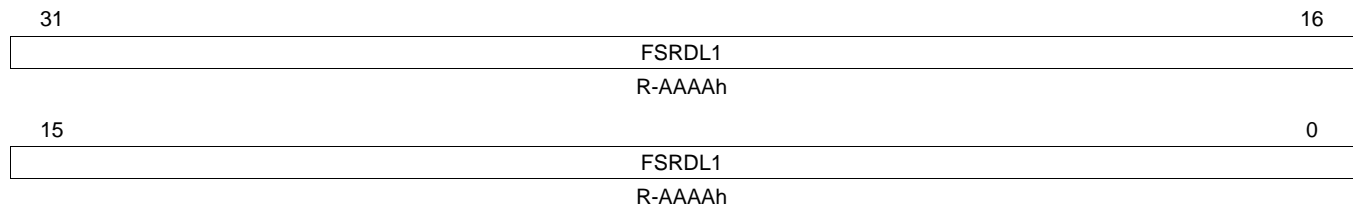


LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-15. Fail Status Data Register 0 (FSRDLO) Field Descriptions**

Bit	Field	Value	Description
31-0	FSRDLO		Failure data on port 0

**Figure 7-15. Fail Status Data Register 1 (FSRDL1) [offset = 01B0h]**



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

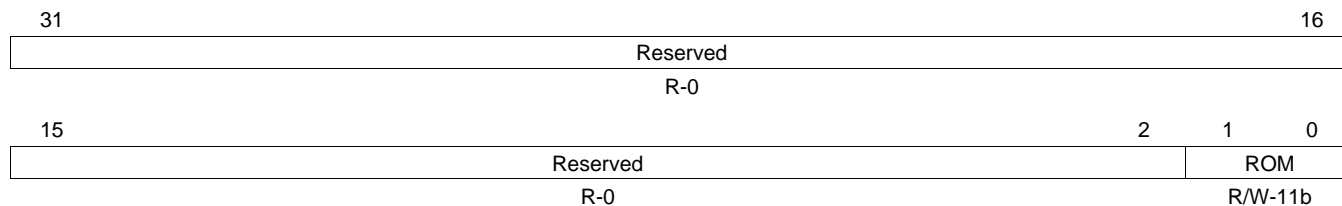
**Table 7-16. Fail Status Data Register 1 (FSRDL1) Field Descriptions**

Bit	Field	Value	Description
31-0	FSRDL1		Failure data on port 1

### 7.5.10 ROM Mask Register (ROM)

This two-bit register sets appropriate ROM access modes for the PBIST controller. The default value is 11b. This register is illustrated in [Figure 7-16](#). It can be programmed according to [Table 7-17](#).

**Figure 7-16. ROM Mask Register (ROM) [offset = 01C0h]**



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

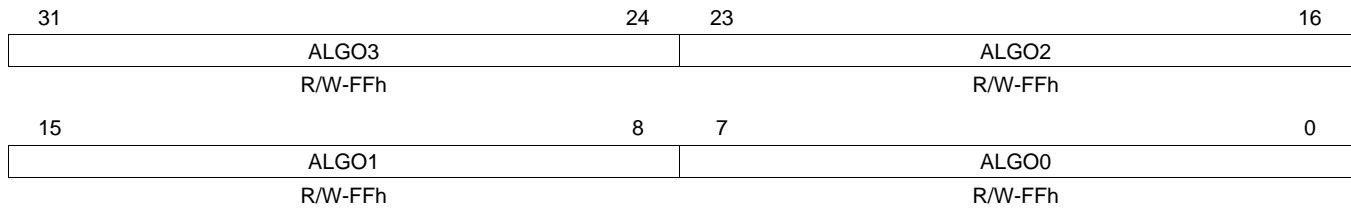
**Table 7-17. ROM Mask Register (ROM) Field Descriptions**

Bit	Field	Value	Description
31-2	Reserved	0	Read returns 0. Writes have no effect.
1-0	ROM	0	No information is used from ROM
		1h	Only RAM Group information from ROM
		2h	Only Algorithm information from ROM
		3h	Both Algorithm and RAM information from ROM. This option should be selected for application self-test.

### 7.5.11 ROM Algorithm Mask Register (ALGO)

This register is used to indicate the algorithm(s) to be used for the memory self-test routine. Each bit corresponds to a specific algorithm. For example, bit 0 controls whether algorithm 1 is enabled or not. [Figure 7-17](#) and [Table 7-18](#) illustrate this register.

**Figure 7-17. ROM Algorithm Mask Register (ALGO) [offset = 01C4h]**



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

**Table 7-18. Algorithm Mask Register (ALGO) Field Descriptions**

Bit	Field	Value	Description
31		0	Algorithm 32 is not selected
		1	Selects algorithm 32 for PBIST run
30		0	Algorithm 31 is not selected
		1	Selects algorithm 31 for PBIST run
...			
0		0	Algorithm 1 is not selected
		1	Selects algorithm 1 for PBIST run
31-0		0	None of the algorithms are selected

**NOTE:** Please refer to [Table 7-2](#) for available algorithms and the memories on which each algorithm can be run.

### 7.5.12 RAM Info Mask Lower Register (RINFOL)

This register is to select RAM groups to run the algorithms selected in the ALGO register. For an algorithm to be executed on a particular RAM group, the corresponding bit in this register must be set to '1'. The default value of this register is all '1's, which means all the RAM Groups are selected. [Figure 7-18](#) and [Table 7-19](#) illustrate this register.

The information from this register is used only when bit 0 in OVER register is not set.

**Figure 7-18. RAM Info Mask Lower Register (RINFOL) [offset = 01C8h]**

31	24	23	16
RINFOL3		RINFOL2	
R/W-FFh		R/W-FFh	
15	8	7	0
RINFOL1		RINFOL0	
R/W-FFh		R/W-FFh	

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

**Table 7-19. RAM Info Mask Lower Register (RINFOL) Field Descriptions**

Bit	Field	Value	Description
31		0	RAM Group 32 is not selected
		1	Selects group 32 for PBIST run
30		0	RAM Group 31 is not selected
		1	Selects RAM group 31 for PBIST run
...			
0		0	RAM Group 1 is not selected
		1	Selects RAM Group 1 for PBIST run
31-0		0	None of the RAM Groups 1 to 32 are selected

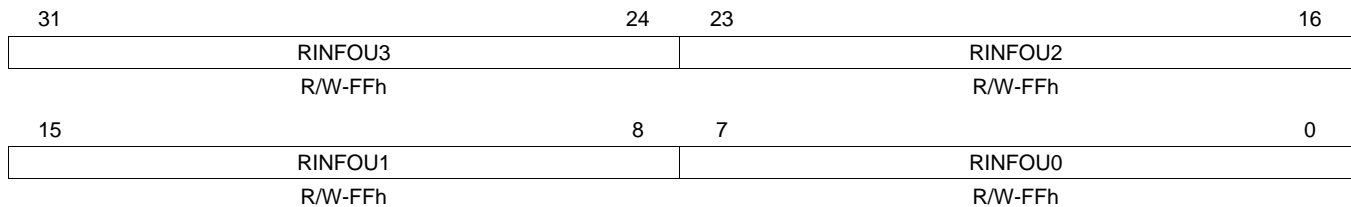
**NOTE:** Please refer to [Table 7-1](#) for RAM info groups.

### 7.5.13 RAM Info Mask Upper Register (RINFOU)

This register is to select RAM groups to run the algorithms selected in the ALGO register. For an algorithm to be executed on a particular RAM group, the corresponding bit in this register should be set to '1'. The default value of this register is all '1's, which means all the RAM Info Groups would be selected.

Figure 7-19 and Table 7-20 illustrate this register.

**Figure 7-19. RAM Info Mask Upper Register (RINFOU) [offset = 01CCh]**



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

**Table 7-20. RAM Info Mask Upper Register (RINFOU) Field Descriptions**

Bit	Field	Value	Description
31		0	RAM Group 64 is not selected
		1	Selects group 64 for PBIST run
30		0	RAM Group 63 is not selected
		1	Selects RAM group 63 for PBIST run
0		0	RAM Group 33 is not selected
		1	Selects RAM Group 33 for PBIST run
31-0		0	None of RAM Groups 33 to 64 are selected

## 7.6 PBIST Configuration Example

The following examples assume that the PLL is locked and selected as clock source with HCLK = 160MHz and VCLK = 80MHz.

### 7.6.1 Example 1 : Configuration of PBIST Controller to Run Self-Test on DCAN1 RAM

This example explains the configurations for running March13, Down1A and Mapcolumn algorithms on DCAN1.

1. Program the HCLK to PBIST ROM clock ratio to 1:2 in System Module.  
MSTGCR[9:8] = 1
2. Enable PBIST Controller in System Module.  
MSIENA[31:0] = 0x00000001
3. Enable the PBIST self-test in System Module.  
MSTGCR[3:0] = 0xA
4. Wait for at least 32 VCLK cycles in a software loop
5. Enable the Pbish internal clocks and ROM interface clock.  
PACT = 0x3
6. Disable RAM Override. This will make the PBIST controller use the information provided by the application in the RINFOx and ALGO registers for the memory self-test.  
OVER = 0x0;
7. Select the Algorithm (Refer [Table 7-2](#) ).  
ALGO = 0x00000054 (Algo 3 = March13N, Algo 5 = down1A\_red, Algo 7 = Map column for two-port DCAN1 RAM)
8. Program the RAM group Info to select DCAN1 (DCAN1 RAM is Group 3, refer [Table 7-1](#))  
RINFOL = 0x00000004 (Select RAM Group 3)  
RINFOU = 0x00000000 (since this device supports only 28 RAM Groups)
9. Select both Algorithm and RAM information from on chip PBIST ROM.  
ROM = 0x3
10. Configure PBIST to run in ROM Mode and start PBIST run.  
DLR = 0x14
11. Wait for PBIST test to complete by polling MSTDONE bit in System Module.  
while (MSTDONE !=1);
12. Once self-test is completed, check the Fail Status registers FSRF0 and FSRF1.  
In case there is a failure (FSRF0 or FSRF1=0x01)
  - (a) Read RAMT register which indicates the RGS and RDS values of the failure RAM.
  - (b) Read FSRC0 and FSRC1 registers which contains the failure count.
  - (c) Read FSRA0 and FSRA1 registers which contains the address of first failure.
  - (d) Read FSRDL0 and FSRDL1 registers which contains the failure data.
  - (e) Resume the Test if required using Program Control register (offset = 0x16C) STR = 2.

In case there is no failure (FSRF0 and FSRF1=0x00) the memory self-test is completed.

- (i) Disable the PBIST internal and ROM clocks.  
PACT = 0
- (ii) Disable the PBIST self-test.  
MSTGCR[3:0] = 0x5

### 7.6.2 Example 2 : Configuration of PBIST Controller to Run Self-Test on ALL RAM Groups

This example explains the configurations for running March13, Down1A and Mapcolumn algorithms on all RAM groups defined in the PBIST ROM.

1. Program the HCLK to PBIST ROM clock ratio to 1:2 in System Module.  
MSTGCR[9:8] = 1
2. Enable PBIST Controller in System Module.

```
MSIENA[31:0] = 0x00000001
```

3. Enable the PBIST self-test in System Module.  
MSTGCR[3:0] = 0xA
  4. Wait for at least 32 VCLK cycles in a software loop.
  5. Enable the Pbish internal clocks and ROM interface clock.  
PACT = 0x3
  6. Enable RAM Override.  
OVER = 0x1;
  7. Select the Algorithms to be run (Refer [Table 7-2](#) )  
ALGO = 0x000000FC (Select March13N, Down1A and Map Column algorithms for single-port and two-port RAMs)
  8. Select both Algorithm and RAM information from on chip PBIST ROM.  
ROM = 0x3
  9. Configure PBIST to run in ROM Mode and kickoff PBIST test.  
DLR = 0x14
  10. Wait for PBIST test to complete by polling MSTDONE bit in System Module.  
while(MSTDONE !=1);
  11. Once self-test is completed, check the Fail Status registers FSRF0 and FSRF1.  
In case there is a failure (FSRF0 or FSRF1 = 0x01)
    - (a) Read RAMT register which indicates the RGS and RDS values of the failure RAM.
    - (b) Read FSRC0 and FSRC1 registers which contains the failure count.
    - (c) Read FSRA0 and FSRA1 registers which contains the address of first failure.
    - (d) Read FSRDL0 and FSRDL1 registers which contains the failure data.
    - (e) Resume the Test if required using Program Control register (offset = 0x16C) STR = 2.
- In case there is no failure (FSRF0 and FSRF1 = 0x00) the Memory self-test is completed.
- (i) Disable the PBIST internal and ROM clocks.  
PACT = 0
  - (ii) Disable the PBIST self-test.  
MSTGCR[3:0] = 0x5





## **CPU Self-Test Controller (STC) Module**

---

---

---

This chapter describe the basics and configuration of the CPU self-test controller present in the device.

<b>Topic</b>	<b>Page</b>
<b>8.1 General Description .....</b>	<b>344</b>
<b>8.2 Application Self-test Flow .....</b>	<b>346</b>
<b>8.3 STC Test Coverage and Duration .....</b>	<b>348</b>
<b>8.4 STC Control Registers .....</b>	<b>349</b>
<b>8.5 STC Configuration Example .....</b>	<b>361</b>

## 8.1 General Description

The CPU self-test Controller (STC) is used to test the ARM-CPU core using the Deterministic Logic Built-In self-test (LBIST) Controller as the test engine. To achieve better coverage for the self-test of complex cores like Cortex-R4, on-chip logic BIST is the preferred solution.

### 8.1.1 CPU Self-Test Controller Features

The CPU self-test controller has the following features:

- Capable of running the complete test as well as running few intervals at a time
  - Ability to continue from the last executed interval (test set) as well as ability to restart from the beginning (first test set)
  - Total of 24 intervals supported in this device
- Complete isolation of the self-tested CPU core from the rest of the system during the self-test run
  - The self-tested CPU core master bus transaction signals are configured to be in idle mode during the self-test run
  - Any master access to the CPU core under self-test (example: DMA access to CPU TCM) will be held till the completion of the self-test
- Ability to capture the failure interval number
- Timeout counter for the CPU self-test run as a fail-safe feature
- Able to read the MISR data (shifted from LBIST controller) of the last executed interval of the self-test run for debugging purposes
- STCCLK determines the self-test execution speed, STC Clock divider (STCCLKDIV) register in system module is used to divide HCLK (system clock) to generate STCCLK

### 8.1.2 STC Block Diagram

STC module provides an interface to the LBIST controller implemented on the core.

The CPU STC is composed of following blocks of logic:

- ROM Interface
- FSM and Sequence control
- Register Block
- Peripheral Bus Interface (VBUSP Interface)
- STC Bypass/ATE interface

#### 8.1.2.1 ROM Interface

This block handles the ROM address and control signal generation to read the self-test microcode from the ROM. The test microcode and golden signature value for each interval are stored in ROM.

##### 8.1.2.1.1 FSM and Sequence Control

This block generates the signals and data to the LBIST controller based on the seed, test\_type and scan chain depth.

##### 8.1.2.1.2 Clock Control

The CLOCK CNTRL sub-block handles the internal clock selection and clock generation for the ROM and LBIST controller.

#### 8.1.2.2 Register Block

This block handles the control of the self-test controller. This block contains various configuration and

status registers which provide the result of a self-test run. These registers are memory mapped and accessible through the Peripheral Bus (VBUSP) Interface. It includes control registers to specify the test model type like stuck-at or transition delay methodologies. This block controls the reseeding (reloading the existing seed of the PRPG) in the LBIST controller. This also handles the buffering of the LBIST seed data and pipelines according to the number of shadow scan chains.

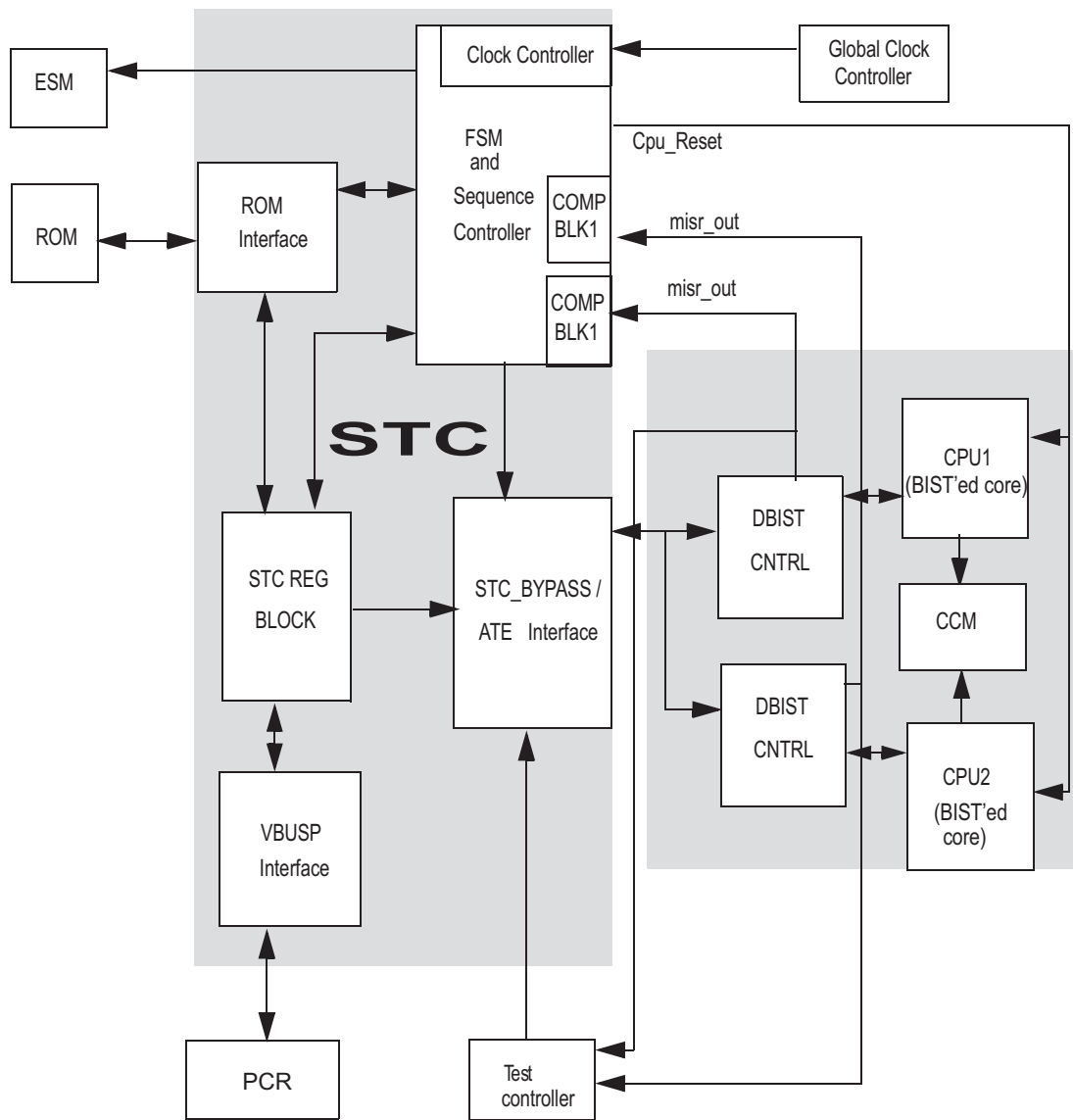
### 8.1.2.3 STC Bypass / ATE Interface

This is a production test interface. Only for TI internal use.

### 8.1.2.4 Peripheral Bus (VBUSP) Interface

STC control registers are accessed through Peripheral Bus (VBUSP) Interface. During application programming, configuration registers are programmed through the Peripheral Bus Interface to enable and run the self-test controller.

Figure 8-1. STC Block Diagram



## 8.2 Application Self-test Flow

This section describes the STC module configuration and the application self-test flow that user should follow for successful execution.

The following two configurations must be part of the startup code:

- STC clock rate configuration, STC clock divider (STCCLKDIV) register in system module is used to divide HCLK(system clock) to generate STCCLK
- Clear SYSESR register before triggering an STC test

### 8.2.1 STC Module Configuration

- Configure the test interval count using STCGCR0[31:16] register. A maximum of 24 intervals are supported in the device. The user can run 24 intervals together or in slices. If the tests are run in slices, the user software can specify to the self-test controller whether to continue the run from the next interval onwards or to restart from interval 0 using bit STCGCR0[0]. This bit gets reset after the completion of the self-test run.
- Configure self-test run timeout counter preload register STCTPR. This register contains the total number of VBUS clock cycles it will take before a self-test timeout error (TIMEOUT\_ERR) will be triggered after the initiation of the self-test run.
- Enable CPU self-test by writing the enable key to STCGCR1 register.

### 8.2.2 Context Saving

STC generates a CPU reset after completion of the test regardless of pass or fail. The user can run the STC test during startup or can divide STC into 24 intervals and run them during normal operation.

If STC is ran only on startup, the user software need not save the CPU contents since the reset caused will go through all startup configurations. User should check the STCGSTAT register for the self-test status before going to the application software.

If STC is divided into intervals and run, user software must save the CPU contents and reload them after the CPU reset caused by the completion of the STC test interval. The check for STC status should bypass STC run if the reset is caused by an STC run to prevent a cyclic reset, i.e., if reset is caused by STC the second time through, then it should not be ran again. The user should also check the STCGSTAT register for the self-test status before restoring the application software.

Following are some of the registers that are required to be backed up before and restored after self-test:

1. CPU core registers (all modes R0-R15, PC, CPSR)
2. CP15 System Control Coprocessor registers - MPU control and configuration registers, Auxiliary Control Register used to Enable ECC, Fault Status Register etc.
3. CP13 Coprocessor Registers - FPU configuration registers, General Purpose Registers
4. Hardware Break Point and watch point registers like BVR, BSR, WVR, WSR etc.

For more information on the CPU reset, please refer to the *Cortex-R4 Technical Reference Manual version r1p3*.

---

**NOTE:** Check all reset source flags in the SYSESR register after a CPU BIST execution. If a flag in addition to CPU reset is set, clear the CPU reset flag and service the other reset sources accordingly.

---

### 8.2.3 Entering CPU Idle Mode

After enabling the STC test by writing the STC enable key, the test is triggered only after the CPU is taken to idle mode by executing the CPU Idle Instruction **asm(" WFI" )**.

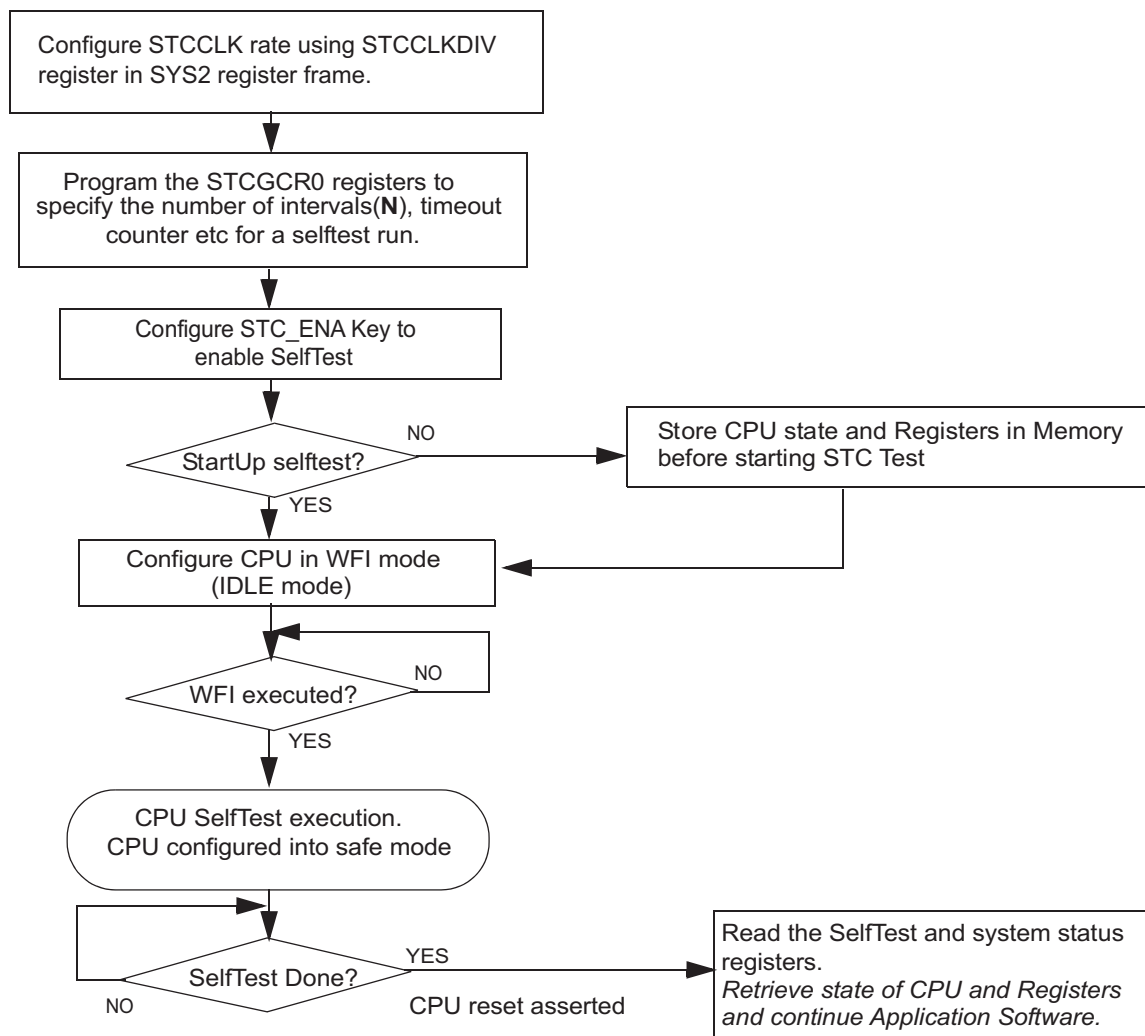
### 8.2.4 Self-Test Completion and Error Generation

At the end of each interval, the 128 bit MISR value (reflected in registers CPU<sub>x</sub>\_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 a CPU self-test, the STC controller updates the status flags in Global Status Register (STCGSTAT) and resets the CPU. In case of an MISR mismatch or a test timeout, an error is generated through the ESM module. TEST\_ERR signal is asserted when an MISR miscompare occurs during the self-test. A TIMEOUT\_ERR is asserted when a timeout 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 are combined and mapped to a single ESM channel. To identify which error occurred, user software must check global status register (STCGSTAT) and fail status register STCFSTAT in the ESM interrupt service routine.

Figure 8-2 illustrates the application self-test test flow chart, drawn based on assumption that the device has gone through startup, necessary clocks initialized and SYSESR register bits cleared.

Figure 8-2. Application Self-Test Flow Chart



### 8.3 STC Test Coverage and Duration

The test coverage and number of test execution cycles (STCCLK) for each test interval when the device is running at HCLK = 180 MHz, VCLK = 90 MHz, and STCCLK = 90 MHz are shown in [Table 8-1](#).

**Table 8-1. STC Test Coverage and Duration**

Intervals	Test Coverage	Test Cycle	Test Time (us)	Intervals	Test Coverage	Test Cycle	Test Time (us)
0	0	0	0	13	86.67	17745	197.17
1	62.13	1365	15.17	14	87.16	19110	212.33
2	70.09	2730	30.33	15	87.61	20475	227.50
3	74.49	4095	45.50	16	87.98	21840	242.67
4	77.28	5460	60.67	17	88.38	23205	257.83
5	79.28	6825	75.83	18	88.69	24570	273.00
6	80.90	8190	91.00	19	88.98	25935	288.17
7	82.02	9555	106.17	20	89.28	27300	303.33
8	83.10	10920	121.33	21	89.50	28665	318.50
9	84.08	12285	136.50	22	89.76	30030	333.67
10	84.87	13650	151.67	23	90.01	31395	348.83
11	85.59	15015	166.83	24	90.21	32760	364.00
12	86.11	16380	182.00				

[Table 8-2](#) gives the typical STC execution times for 24 intervals at different clock rates.

**Table 8-2. Typical STC Execution Times**

Number of Intervals	@ HCLK = 180 MHz VCLK = 90 MHz STCCLK = 90 MHz	@ HCLK = 100 MHz VCLK = 100 MHz STCCLK = 50MHz	@ HCLK = 160 MHz VCLK = 80 MHz STCCLK = 80 MHz
24	364 us	655.20 us	409.50 us

## 8.4 STC Control Registers

STC control registers are accessed through Peripheral Bus (VBUSP) interface. Read and write access in 8,16 and 32 bit are supported. The base address for the control registers is FFFF E600h.

All reserved bits are read as zeros. Write to reserved fields has no effect.

---

**NOTE:** In suspend mode, all registers can be written irrespective of user or privilege mode.

---

**Table 8-3. STC Control Registers**

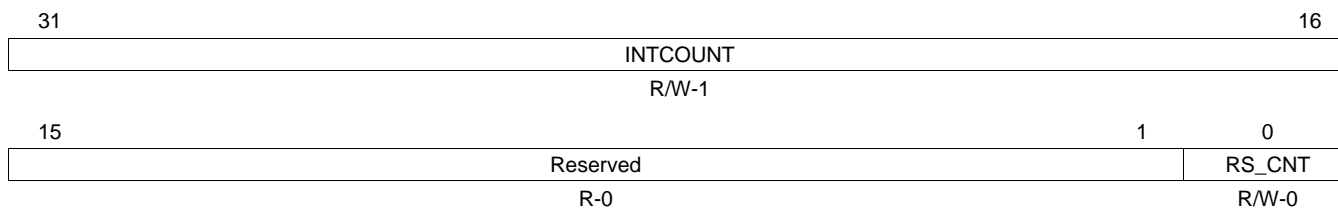
Offset	Acronym	Register Description	Section
00	STCGCR0	STC Global Control Register 0	<a href="#">Section 8.4.1</a>
04h	STCGCR1	STC Global Control Register 1	<a href="#">Section 8.4.2</a>
08h	STCTPR	Self-Test Run Timeout Counter Preload Register	<a href="#">Section 8.4.3</a>
0Ch	STC_CADDR	STC Current ROM Address Register	<a href="#">Section 8.4.4</a>
10h	STCCICR	STC Current Interval Count Register	<a href="#">Section 8.4.5</a>
14h	STCGSTAT	Self-Test Global Status Register	<a href="#">Section 8.4.6</a>
18h	STCFSTAT	Self-Test Fail Status Register	<a href="#">Section 8.4.7</a>
2Ch	CPU1_CURMISR3	CPU1 Current MISR Register	<a href="#">Section 8.4.8</a>
30h	CPU1_CURMISR2	CPU1 Current MISR Register	<a href="#">Section 8.4.8</a>
34h	CPU1_CURMISR1	CPU1 Current MISR Register	<a href="#">Section 8.4.8</a>
38h	CPU1_CURMISR0	CPU1 Current MISR Register	<a href="#">Section 8.4.8</a>
3Ch	CPU2_CURMISR3	CPU2 Current MISR Register	<a href="#">Section 8.4.9</a>
40h	CPU2_CURMISR2	CPU2 Current MISR Register	<a href="#">Section 8.4.9</a>
44h	CPU2_CURMISR1	CPU2 Current MISR Register	<a href="#">Section 8.4.9</a>
48h	CPU2_CURMISR0	CPU2 Current MISR Register	<a href="#">Section 8.4.9</a>



### 8.4.1 STC Global Control Register0 (STCGCR0)

This register is described in [Figure 8-3](#) and [Table 8-4](#). The offset address is 00.

**Figure 8-3. STC Global Control Register0 (STCGCR0)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after nPORST (power on reset) or System reset

**Table 8-4. STC Global Control Register0 (STCGCR0) Field Descriptions**

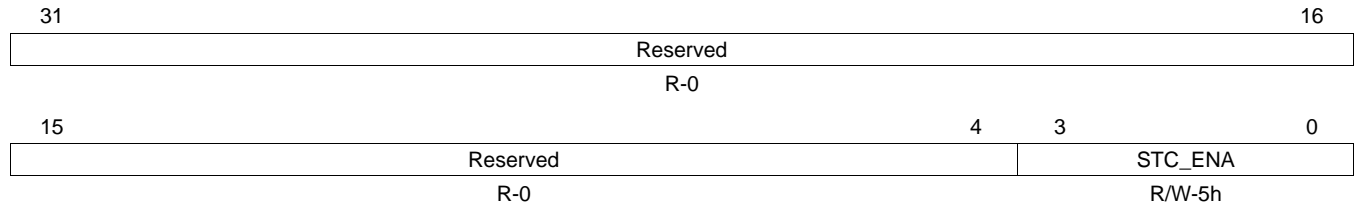
Bit	Field	Value	Description
31-16	INTCOUNT		Number of intervals of self-test run This register specifies the number of intervals to run for the self-test run. This corresponds to the number of intervals to be ran from the value reflected in the current interval counter.
15-1	Reserved	0	Read returns 0. Writes have no effect.
0	RS_CNT		Restart or Continue This bit specifies whether to continue the run from next interval onwards or to restart from interval 0. This bit gets reset after the completion of a self-test run.
		0	Continue STC run from the previous interval.
		1	Restart STC run from interval 0.

**NOTE:** On a powerup reset or system reset this register gets reset to its default values.

### 8.4.2 STC Global Control Register1 (STCGCR1)

This register is described in [Figure 8-4](#) and [Table 8-5](#). The offset address is 04.

**Figure 8-4. STCGCR1 (STC Global Control Register1)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after nPORST (power on reset) or System reset

**Table 8-5. STCGCR1 (STC Global Control Register1) Field Descriptions**

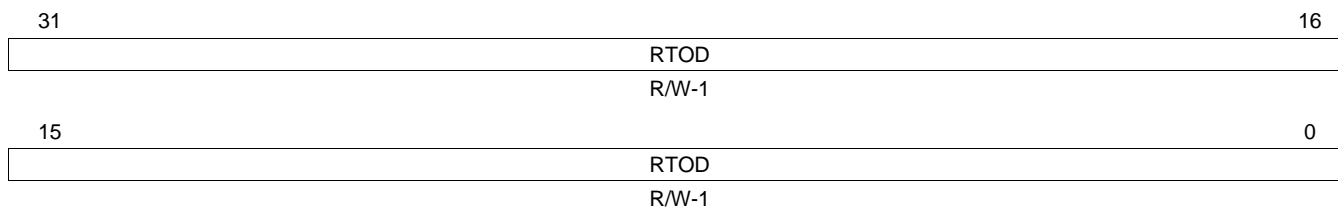
Bit	Field	Value	Description
31-4	Reserved	0	Read returns 0. Writes have no effect.
3-0	STC_ENA	Ah	Self-test run enable key Self-test run enabled.
		All Others	Self-test run disabled.

**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.

### 8.4.3 Self-Test Run Timeout Counter Preload Register (STCTPR)

This register is described in [Figure 8-5](#) and [Table 8-6](#). The offset address is 08h.

**Figure 8-5. Self-Test Run Timeout Counter Preload Register (STCTPR)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after nPORST (power on reset) or System reset

**Table 8-6. Self-Test Run Timeout Counter Preload Register (STCTPR)**

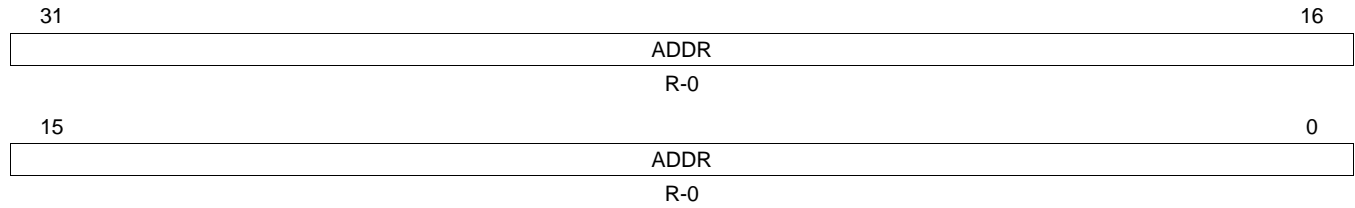
Bit	Field	Value	Description
31-0	RTOD		<p>Self-test timeout count preload</p> <p>This register contains the total number of VBUS clock cycles it will take before a self-test timeout error (TIMEOUT_ERR) will be triggered after the initiation of the self-test run. This is a fail safe feature to not hang-up the system on account of any run away self-test issues.</p> <p>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.</p>

**NOTE:** This register gets reset to its default value with Power on or system reset assertion.

#### 8.4.4 STC Current ROM Address Register (STC\_CADDR)

This register is described in [Figure 8-6](#) and [Table 8-7](#). The offset address is 0Ch.

**Figure 8-6. STC Current ROM Address Register (STC\_CADDR)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after nPORST (power on reset) or System reset

**Table 8-7. STC Current ROM Address Register (STC\_CADDR) Field Descriptions**

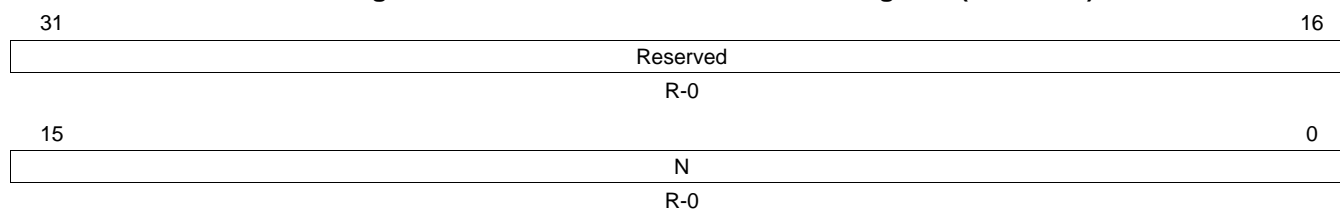
Bit	Field	Value	Description
31-0	ADDR		Current ROM Address This register reflects the current ROM address (for micro code load) which is the current value of the STC program counter.

**NOTE:** When the 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.

### 8.4.5 STC Current Interval Count Register (STCCICR)

This register is described in [Figure 8-7](#) and [Table 8-8](#). The offset address is 10h.

**Figure 8-7. STC Current Interval Count Register (STCCICR)**



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

**Table 8-8. STC Current Interval Count Register (STCCICR) Field Descriptions**

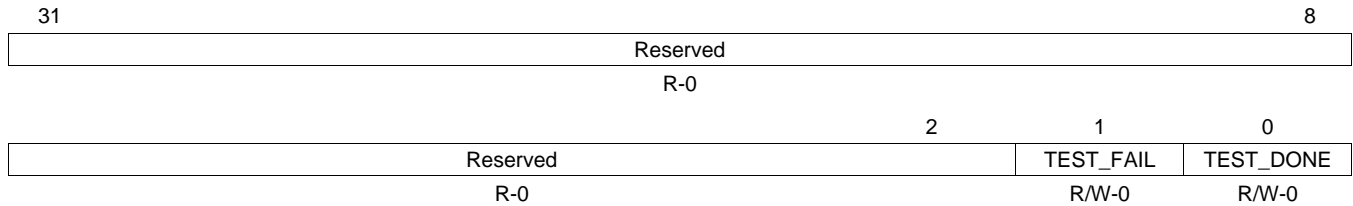
Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	N		Interval Number This specifies the last executed interval number.

**NOTE:** When the global configuration register restart bit STCGCR[0] is set to a 1 or on a powerup reset the Current interval counter resets to default value.

### 8.4.6 Self-Test Global Status Register (STCGSTAT)

This register is described in [Figure 8-8](#) and [Table 8-9](#). The offset address is 14h.

**Figure 8-8. Self-Test Global Status Register (STCGSTAT)**



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

**Table 8-9. Self-Test Global Status Register (STCGSTAT) Field Descriptions**

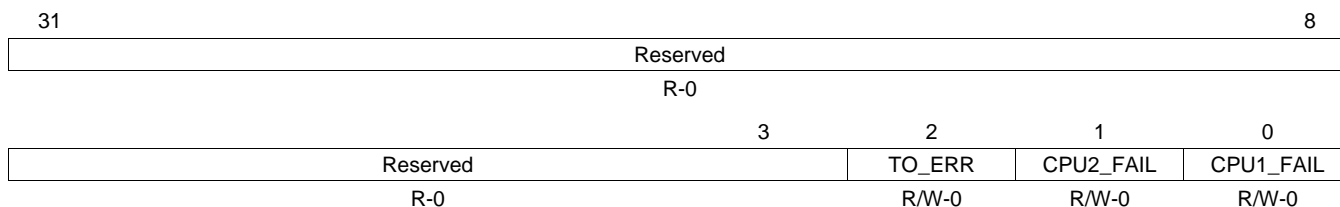
Bit	Field	Value	Description
31-2	Reserved	0	Read returns 0. Writes have no effect.
1	TEST_FAIL	0	Self-test run has not failed.
		1	Self-test run has failed.
0	TEST_DONE	0	Not completed.
		1	Self-test run completed. The test done flag is set to a 1 for any of the following conditions: <ol style="list-style-type: none"> <li>1. When the STC run is complete without any failure</li> <li>2. When a failure occurs on a STC run</li> <li>3. When a timeout failure occurs</li> </ol> Reset is generated to the CPU on which the STC run is being performed when TEST_DONE goes high (the test is completed).

**NOTE:** The above 2 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 2 status flags gets cleared to their default values. This register gets reset to its default value with power on reset assertion.

### 8.4.7 Self-Test Fail Status Register (STCFSTAT)

This register is described in [Figure 8-9](#) and [Table 8-10](#). The offset address is 18h.

**Figure 8-9. Self-Test Fail Status Register (STCFSTAT)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after nPORST (power on reset) or System reset

**Table 8-10. Self-Test Fail Status Register (STCFSTAT) Field Descriptions**

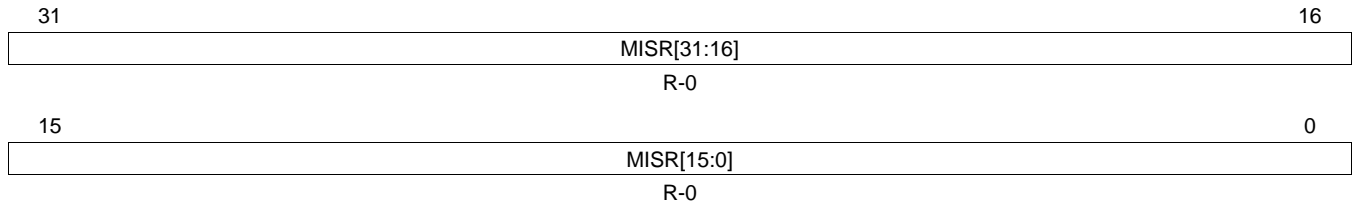
Bit	Field	Value	Description
31-3	Reserved	0	Read returns 0. Writes have no effect.
2	TO_ERR	0	No time out error occurred.
		1	Self-test run failed due to a timeout error.
1	CPU2_FAIL	0	CPU2 failure info No MISR mismatch for CPU2.
		1	Self-test run failed due to MISR mismatch for CPU2.
0	CPU1_FAIL	0	CPU1 failure info 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 of 1. Additionally when the STC\_ENA Key in STCGCR1 (is enabled from a non Ah value to a Ah value the above 3 status bits get cleared to their default values. This register gets reset to its default value with power on reset assertion.

**8.4.8 CPU1 Current MISR Register (CPU1\_CURMISR[3:0])**

This register is described in [Figure 8-10](#) through [Figure 8-13](#) and [Table 8-11](#). The offset address for CPU1\_CURMISR3 is 2Ch.

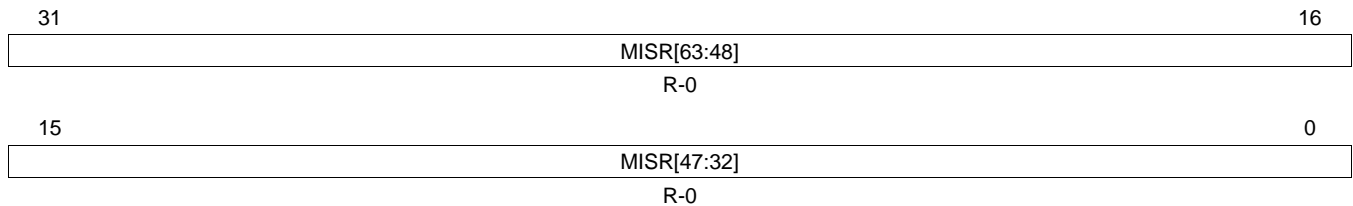
**Figure 8-10. CPU1 Current MISR Register (CPU1\_CURMISR3)**



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

The offset address for CPU1\_CURMISR2 is 30h.

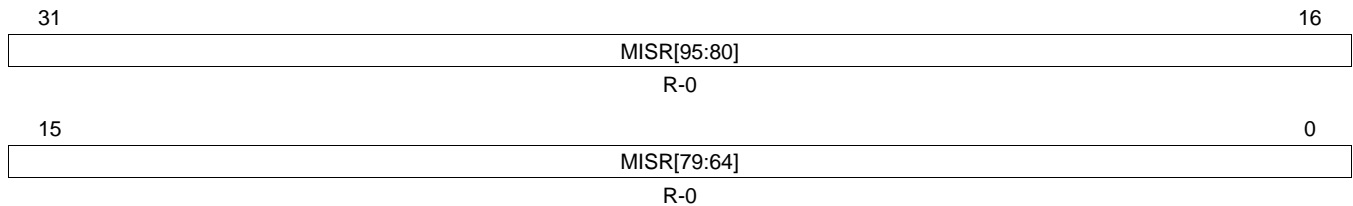
**Figure 8-11. CPU1 Current MISR Register (CPU1\_CURMISR2)**



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

The offset address for CPU1\_CURMISR1 is 34h.

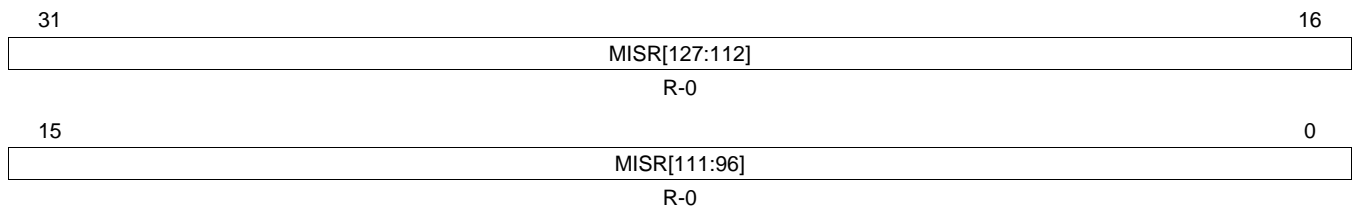
**Figure 8-12. CPU1 Current MISR Register (CPU1\_CURMISR1)**



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

The offset address for CPU1\_CURMISR0 is 38h.

**Figure 8-13. CPU1 Current MISR Register (CPU1\_CURMISR0)**



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

**Table 8-11. CPU1 Current MISR Register (CPU1\_CURMISR[3:0]) Field Descriptions**

Bit	Field	Value	Description
127-0	MISR		MISR data from CPU1 This register contains the MISR data from the CPU1 for the most recent interval. This value is compared with the GOLDEN MISR value copied from ROM.



---

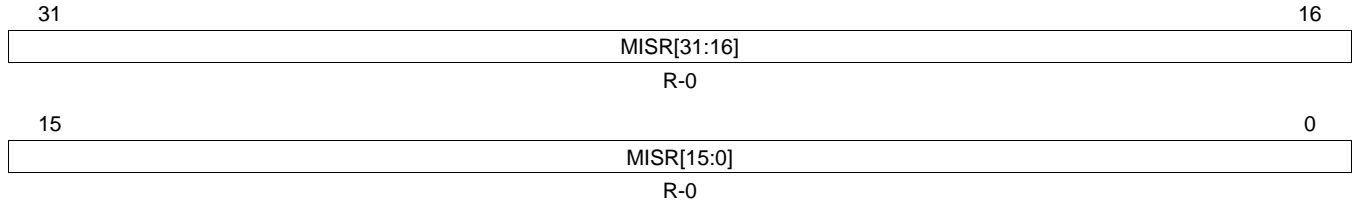
**NOTE:** This register gets reset to its default value with power on or system reset assertion.

---

**8.4.9 CPU2\_CURMISR[3:0] (CPU2 Current MISR Register)**

This register is described in [Figure 8-14](#) through [Figure 8-17](#) and [Table 8-12](#). The offset address for CPU2\_CURMISR3 is 3Ch.

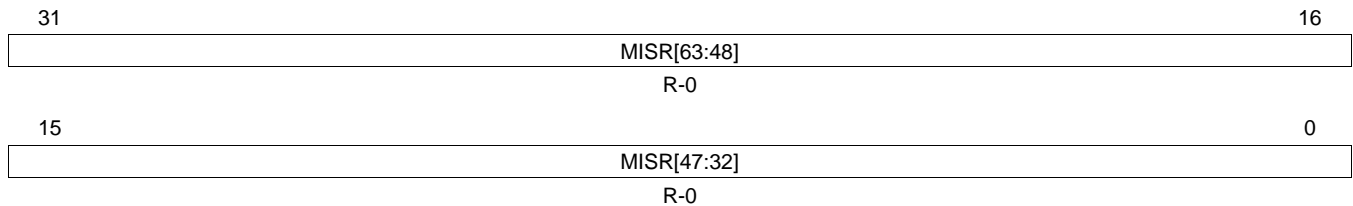
**Figure 8-14. CPU2 Current MISR Register (CPU2\_CURMISR3)**



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

The offset address for CPU2\_CURMISR2 is 40h.

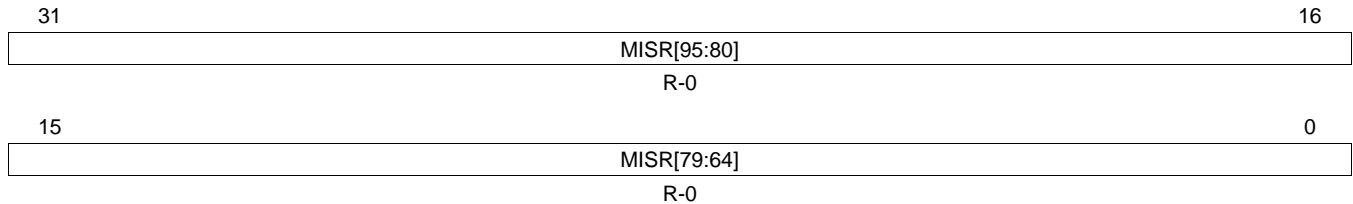
**Figure 8-15. CPU2 Current MISR Register (CPU2\_CURMISR2)**



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

The offset address for CPU2\_CURMISR1 is 44h.

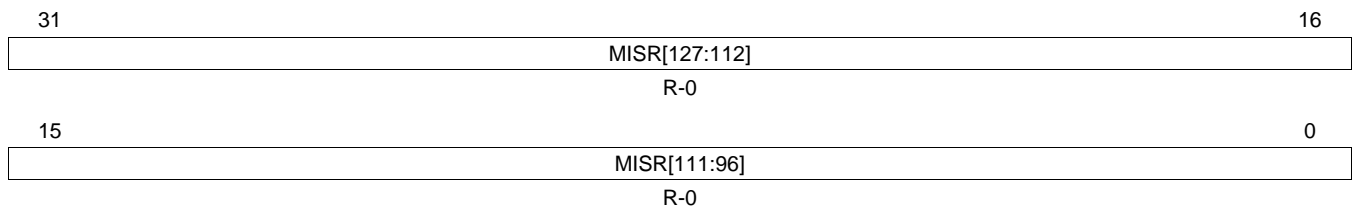
**Figure 8-16. CPU2 Current MISR Register (CPU2\_CURMISR1)**



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

The offset address for CPU2\_CURMISR0 is 48h.

**Figure 8-17. CPU2 Current MISR Register (CPU2\_CURMISR0)**



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

**Table 8-12. CPU2 Current MISR Register (CPU2\_CURMISR[3:0]) Field Descriptions**

Bit	Field	Value	Description
127-0	MISR		MISR data from CPU2 This register contains the MISR data from the CPU2 for the most recent interval. This value is compared with the GOLDEN MISR value copied from ROM.

---

**NOTE:** This register gets reset to its default value with power on or system reset assertion.

---

## 8.5 STC Configuration Example

The following examples assume that the PLL is locked and selected as the system clock source with HCLK = 180 MHz and VCLK = 90 MHz.

### **Example 8-1. Self-test Run for 24 Interval**

This example explains the configurations for running STC Test for maximum Test Intervals 24

1. Maximum STC clock rate support at 180 MHz HCLK is 90 MHz. Divide HCLK by 2 to achieve this clock rate. STCCLKDIV[26:24] register in the secondary system module frame at location 0xFFFF E108 is used. STCCLKDIV[26:24] = 1
2. Clear CPU RST status bit in System Exception Status Register in the system Module. SYSESR[5] = 1
3. Configure the Test Interval count in STC module. STCGCR0[31:16] = 24
4. Configure self-test Run Time out counter preload Register. STCTPR[31:0] = 0xFFFFFFFF
5. Enable CPU self-test. STCGCR1[3:0] = 0xA;
6. Perform a context save of CPU state and configuration registers that gets reset on CPU reset.
7. Configure the CPU in idle mode by executing CPU Idle Instruction. **asm(" WFI")**
8. Upon CPU reset verify, the CPU RST status bit in System Exception Status Register is set. This also verifies that no other resets occurred during the self-test. SYSESR[5] == 1
9. Check the STCGSTAT register for the self-test Status. Check TEST\_DONE bit before evaluating TEST\_FAIL bit.

If TEST\_DONE =0 the self-test is not completed. Restart the STC test.

If (TEST\_DONE =1 and TEST\_FAIL= 1) the self-test is completed and Failed.

- Read STC Fail Status Register STCFSTAT[2:0] to identify the type of Failure (Timeout, CPU1 fail, CPU2 fail).

In case there is no failure (TEST\_DONE =1 and TEST\_FAIL = 0), the Memory self-test is completed successfully.

- Recover the CPU status, configuration registers and continue application software.



## ***CPU Compare Module for Cortex™-R4F (CCM-R4F)***

This device implements two instances of the Cortex-R4F CPU which are running in lockstep to detect faults which may result in unsafe operating conditions. The CCM-R4F detects faults and signals them to the ESM.

Topic	Page
<b>9.1 Main Features .....</b>	<b>364</b>
<b>9.2 Block Diagram .....</b>	<b>364</b>
<b>9.3 Module Operation .....</b>	<b>365</b>
<b>9.4 Control Registers .....</b>	<b>368</b>

## 9.1 Main Features

Safety-critical applications require run-time detection of faults in the Central Processing Unit (CPU). The CPU Compare Module for Cortex-R4F (CCM-R4F) compares the core compare bus outputs of two Cortex-R4F CPUs running in a 1oo1D lockstep configuration. Any difference in the core compare bus outputs of the CPUs is flagged as an error. For diagnostic purposes the CCM-R4F also incorporates a self test capability to allow for boot time checking of hardware faults within the CCM-R4F itself.

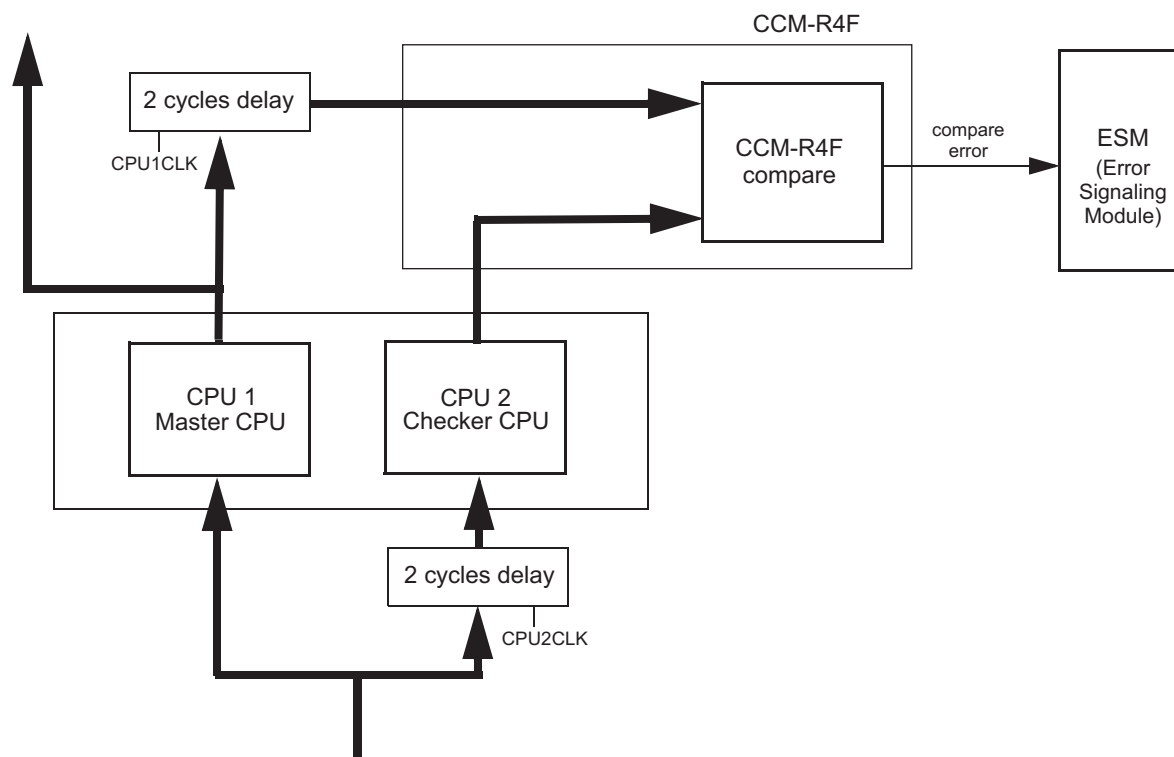
The main features of the CCM-R4F are:

- run-time detection of faults
- self-test capability
- error forcing capability

## 9.2 Block Diagram

Figure 9-1 shows the interconnection diagram of the CCM-R4F with the two Cortex-R4F CPUs. The core compare bus outputs of the CPUs are compared in the CCM-R4F. To avoid common mode impacts, the signals of the CPUs to be compared are temporally diverse. The output signals of the master CPU are delayed 2 cycles while the input signals of checker CPU are delayed 2 cycles.

**Figure 9-1. Block Diagram**



### 9.3 Module Operation

The CCM-R4F compares the core compare bus outputs of the master and checker Cortex-R4F CPUs on the TMS570 microcontroller and signals an error on any mismatch. This comparison is started 6 CPU clock cycles after the CPU comes out of reset to ensure that CPU output signals have propagated to a known value after reset.

The CCM-R4F can run in one out of four operating modes:

1. 1001D lock step,
2. self-test,
3. error forcing, and
4. self-test error forcing

The operating mode can be selected by writing a dedicated key to the key register (MKEY).

#### 9.3.1 1001D Lock Step Mode

This is the default mode on startup.

In the lock step mode, the compare bus output signals of both CPUs are compared. A difference in the CPU compare bus outputs is indicated by ESM error flag “CCM-R4F - compare”.

Not all internal registers of the Cortex R4F CPU have fixed values upon reset. To avoid an erroneous CCMR4F compare error, the application software needs to ensure that the CPU registers of both CPUs are initialized with the same values before the registers are used, including function calls where the register values are pushed onto the stack.

#### 9.3.2 Self-Test Mode

In self test mode the CCM-R4F itself is checked for faults. During self test, the compare error module output signal is deactivated. Any fault detected inside the CCM-R4F will be flagged by ESM error “CCM-R4F - selftest”.

In self test mode, the CCM-R4F automatically generates test patterns to look for any hardware faults. If a fault is detected, then a self test error flag is set, a self test error signal is asserted and sent to the ESM, and the self test is terminated immediately. If no fault is found during self test, the self test complete flag is set. In both cases, the CCM-R4F remains in self test mode after the test has been terminated or completed, and the application needs to switch the CCM-R4F mode by writing another key to the mode key register (MKEY). During the self test operation, the compare error signal output to the ESM is inactive irrespective of the compare result.

There are two types of patterns generated by CCM-R4F during self-test mode:

- (i) Compare Match Test
- (ii) Compare Mismatch Test

CCM-R4F first generates Compare Match Test patterns, followed by Compare Mismatch Test patterns. Each test pattern is applied on both CPU signal inputs of the CCM-R4F's compare block and clocked for one cycle. The duration of self test is 3615 CPU clock cycles (GCLK).

---

**NOTE:** During self test, both CPUs can execute normally, but the compare logic will not be checking any CPU signals. Also during self test, only the compare unit logic is tested and not the memory mapped register controls for the CCM-R4F. The self test is not interruptible.

---

##### 9.3.2.1 Compare Match Test

During the Compare Match Test, there are four different test patterns generated to stimulate the CCM-R4F. An identical vector is applied to both input ports at the same time expecting a compare match. These patterns cause the self test logic to exercise every CPU compare bus output signal in parallel. If the compare unit produces a compare mismatch then the self test error flag is set, the self test error signal is generated, and the Compare Match Test is terminated.



The four test patterns used for the Compare Match Test are:

- All 1s on both CPU signal ports
- All 0s on both CPU signal ports
- 0xAs on both CPU signal ports
- 0x5s on both CPU signal ports

These four test patterns will take four clock cycles to complete. illustrates the sequence of Compare Match Test.

**Table 9-1. Compare Match Test Sequence**

CPU 1 Signal Position									CPU 2 Signal Position									Cycle
n:8	7	6	5	4	3	2	1	0	n:8	7	6	5	4	3	2	1	0	
1s	1	1	1	1	1	1	1	1	1s	1	1	1	1	1	1	1	1	0
0s	0	0	0	0	0	0	0	0	0s	0	0	0	0	0	0	0	0	1
0xA	1	0	1	0	1	0	1	0	0xA	1	0	1	0	1	0	1	0	2
0x5	0	1	0	1	0	1	0	1	0x5	0	1	0	1	0	1	0	1	3

### 9.3.2.2 Compare Mismatch Test

During the Compare Mismatch Test, the number of test patterns is equal to twice the number of CPU output signals to compare in lock step mode. An all 1s vector is applied to the CCM-R4F's CPU1 input port and the same pattern is also applied to the CCM-R4F's CPU2 input port but with one bit flipped starting from signal position 0. The un-equal vector will cause the CCM-R4F to expect a compare mismatch at signal position 0, if the CCM-R4F logic is working correctly. If, however, the CCM-R4F logic reports a compare match, the self test error flag is set, the self test error signal is asserted, and the Compare Mismatch Test is terminated.

This Compare Mismatch Test algorithm repeats in a domino fashion with the next signal position flipped while forcing all other signals to logic level 1. This sequence is repeated until every single signal position is verified on both CPU signal ports.

The Compare Mismatch Test is terminated if the CCM-R4F reports a compare match versus the expected compare mismatch. This test ensures that the compare unit is able to detect a mismatch on every CPU signal being compared. [Table 9-2](#) illustrates the sequence of Compare Mismatch Test.

**Table 9-2. Compare Mismatch Test Sequence**

CPU 1 Signal Position										CPU 2 Signal Position										Cycle
n	n-1:8	7	6	5	4	3	2	1	0	n	n-1:8	7	6	5	4	3	2	1	0	
1	1	1s	1	1	1	1	1	1	1	1	1	1s	1	1	1	1	1	1	0	0
1	1	1s	1	1	1	1	1	1	1	1	1	1s	1	1	1	1	1	0	1	1
1	1	1s	1	1	1	1	1	1	1	1	1	1s	1	1	1	1	0	1	1	2
1	1	1s	1	1	1	1	1	1	1	1	1	1s	1	1	1	0	1	1	1	3
::																				
1	1	1s	1	1	1	1	1	1	1	1	0	1s	1	1	1	1	1	1	1	n-1
1	1	1s	1	1	1	1	1	1	1	0	1	1s	1	1	1	1	1	1	1	n
1	1	1s	1	1	1	1	1	1	0	1	1	1s	1	1	1	1	1	1	1	n+1
1	1	1s	1	1	1	1	1	0	1	1	1	1s	1	1	1	1	1	1	1	n+2
1	1	1s	1	1	1	1	0	1	1	1	1	1s	1	1	1	1	1	1	1	n+3
1	1	1s	1	1	1	0	1	1	1	1	1	1s	1	1	1	1	1	1	1	n+4
::																				
1	0	1s	1	1	1	1	1	1	1	1	1	1s	1	1	1	1	1	1	1	2n-1
0	1	1s	1	1	1	1	1	1	1	1	1	1s	1	1	1	1	1	1	1	2n

### 9.3.3 Error Forcing Mode

In the error forcing mode, a test pattern is applied to the CPU related inputs of the CCM-R4F compare logic to force an error in the compare error output signal of the compare unit. The ESM error flag “CCM-R4F - compare” is expected after the error forcing mode completes.

Error forcing mode is similar to the Compare Mismatch Test operation of self test mode in which an un-equal vector is applied to the CCM-R4F CPU signal ports. Instead of setting a self test error flag and asserting the self test error signal to ESM, the error forcing mode forces the compare mismatch to actually assert the compare error output signal. This ensures that faults in the path between CCM-R4F and ESM is detected.

Only one hardcoded test pattern is applied into CCM-R4F during error forcing mode. A repeated 0x5 pattern is applied to CPU1 signal port of CCM-R4F input while a repeated 0xA pattern is applied to the CPU2 signal port of CCM-R4F input. The error forcing mode takes one cycle to complete. Hence, the failing signature is presented for one clock cycle. After that, the mode is automatically switched to lock step mode. The key register (MKEY) will indicate the lock step key mode once it is switched to lock step mode. During the one cycle required by the error forcing test, the CPU output signals are not compared. User should expect the ESM to trigger a response (report the CCM-R4F fail). If no error is detected by ESM, then a hardware fault is present.

### 9.3.4 Self-Test Error Forcing Mode

In self-test error forcing mode an error is forced at the self-test error signal. The compare unit is still running in lockstep mode and the key is switched to lockstep after one clock cycle. The ESM error flag “CCM-R4F - selftest” is expected after the self-test error forcing mode completes.

### 9.3.5 Operation During CPU Debug Mode

Certain debug operations place the CPU in a halting debug state where the code execution is halted. Because halting debug events are asynchronous, there is a possibility for the debug requests to cause loss of lockstep. CCM-R4F will disable upon detection of halting debug requests. Core compare error will not be generated and flags will not update. A CPU reset is needed to ensure the CPUs are again in lockstep and will also re-enable the CCM-R4F.

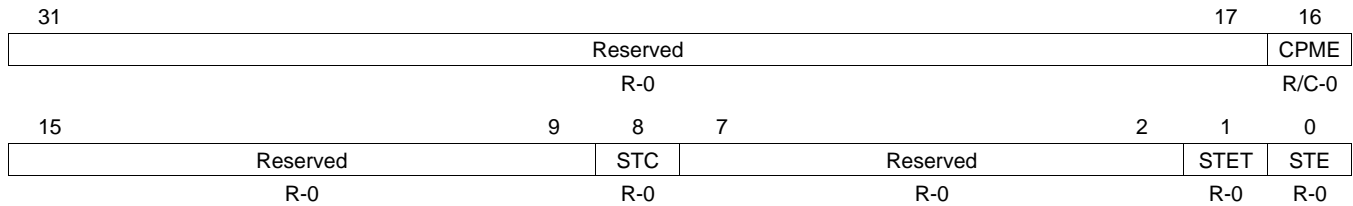
## 9.4 Control Registers

This section describes the CCM-R4F registers. Each register begins on a 32-bit word boundary. The registers support 32-bit, 16-bit and 8-bit accesses.

**Table 9-3. Module Registers**

Offset	Acronym	Register Description	Section
FFFF F600h	CCMSR	CCM-R4F Status Register	<a href="#">Section 9.4.1</a>
FFFF F604h	CCMKEYR	CCM-R4F Key Register	<a href="#">Section 9.4.2</a>

### 9.4.1 CCM-R4F Status Register (CCMSR)

**Figure 9-2. CCM-R4F Status Register (CCMSR) (Offset = FFFF F600h)**


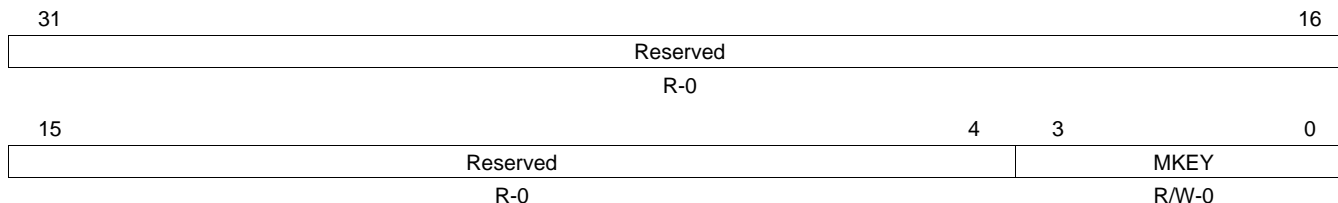
LEGEND: R/W = Read/Write; R = Read only; C = Clear; -n = value after reset

**Table 9-4. CCM-R4F Status Register (CCMSR) Field Descriptions**

Bit	Field	Value	Description
31-17	Reserved	0	Reads return zeros and writes have no effect.
16	CPME	0	Compare Error <b>User and privileged mode:</b> Read: CPU signals are identical.
		1	<b>Privileged mode:</b> Write: Leaves the bit unchanged. <b>User and privileged mode:</b> Read: CPU signal compare mismatch.
			<b>Privileged mode:</b> Write: Clears the bit.
15-9	Reserved		Reads return zeros and writes have no effect.
8	STC	0	Self-test Complete <b>Note:</b> This bit is always 0 when not in self-test mode. Once set, switching from self-test mode to other modes will clear this bit. <b>User and privileged mode:</b> Read: Self-test on-going if self-test mode is entered.
		1	Write: Writes have no effect. <b>User and privileged mode:</b> Read: Self-test is complete.
			Write: Writes have no effect.
7-2	Reserved		Reads return zeros and writes have no effect.
1	STET	0	Self-test Error Type <b>User and privileged mode:</b> Read: Self-test failed during Compare Match Test is STE=1.
		1	Write: Writes have no effect. <b>User and privileged mode:</b> Read: Self-test failed during Compare Match Test is STE=1.
			Write: Writes have no effect.
0	STE	0	Self-test Error <b>Note:</b> This bit gets updated when the self-test is complete or an error is detected. <b>User and privileged mode:</b> Read: Self-test passed.
		1	Write: Writes have no effect. <b>User and privileged mode:</b> Read: Self-test failed.
			Write: Writes have no effect.

### 9.4.2 CCM-R4F Key Register (CCMKEYR)

Figure 9-3. CCM-R4F Key Register (CCMKEYR) (Offset = FFFF F604h)



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

Table 9-5. CCM-R4F Key Register (CCMKEYR) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reads return to zeros and writes have no effect.
3-0	MKEY	0	<b>Mode Key</b> <b>User and privileged mode:</b> Read: Returns current value of the MKEY. <b>Privileged mode:</b> Write: Lockstep mode.
		6h	<b>User and privileged mode:</b> Read: Returns current value of the MKEY. <b>Privileged mode:</b> Write: Self-test mode.
		9h	<b>User and privileged mode:</b> Read: Returns current value of the MKEY. <b>Privileged mode:</b> Write: Error Forcing mode.
		Fh	<b>User and privileged mode:</b> Read: Returns current value of the MKEY. <b>Privileged mode:</b> Write: Self-test Error Forcing mode. <b>Note:</b> It is recommended writing no other keys than the keys listed. Invalid keys will result in switching operation to lockstep mode.

## Oscillator and PLL

This chapter describes the oscillator and PLL clock source paths for the device.

Topic	Page
<b>10.1 Introduction</b> .....	<b>372</b>
<b>10.2 Quick Start</b> .....	<b>373</b>
<b>10.3 Oscillator</b> .....	<b>374</b>
<b>10.4 Low Power Oscillator and Clock Detect (LPOCLKDET)</b> .....	<b>376</b>
<b>10.5 PLL</b> .....	<b>379</b>
<b>10.6 Control Registers</b> .....	<b>386</b>
<b>10.7 Phase-Locked Loop Theory of Operation</b> .....	<b>403</b>
<b>10.8 Programming Example</b> .....	<b>406</b>

## 10.1 Introduction

This Technical Reference Manual chapter provides an overview of the oscillator and PLL clock source paths for the device.

The oscillator macro will pass a signal driven into the OSCIN pin to clock source 0 which is the default clock source on reset. When a crystal or resonator with appropriate load circuitry is connected to OSCIN and OSCOUT, the oscillator macro drives the crystal/resonator to generate the input waveform. In addition to being directly usable as clock source 0, the oscillator clock is the input to the PLL.

The oscillator frequency is continuously monitored by a dedicated clock detect circuit. If the frequency falls out of a fixed range, the clock detect switches the clock from the oscillator to an internally generated, free-running frequency (from the low power oscillator (LPO)).

The phase lock loop (PLL), a circuit in the microcontroller, is used to multiply the input frequency to some higher (device operation) frequency. This frequency synthesis is useful for generating higher frequencies than can be conveniently achieved with an external crystal or resonator. Additionally, the PLL allows the flexibility to be able to synthesize one of multiple frequency options from a given crystal or resonator.

Frequency modulation can be superimposed on the synthesized frequency. The modulation provides a means to reduce the impact of electromagnetic radiation from the device; this reduction in measured radiation can be useful in sensitive applications.

### 10.1.1 Features

The main features of the source clock path are:

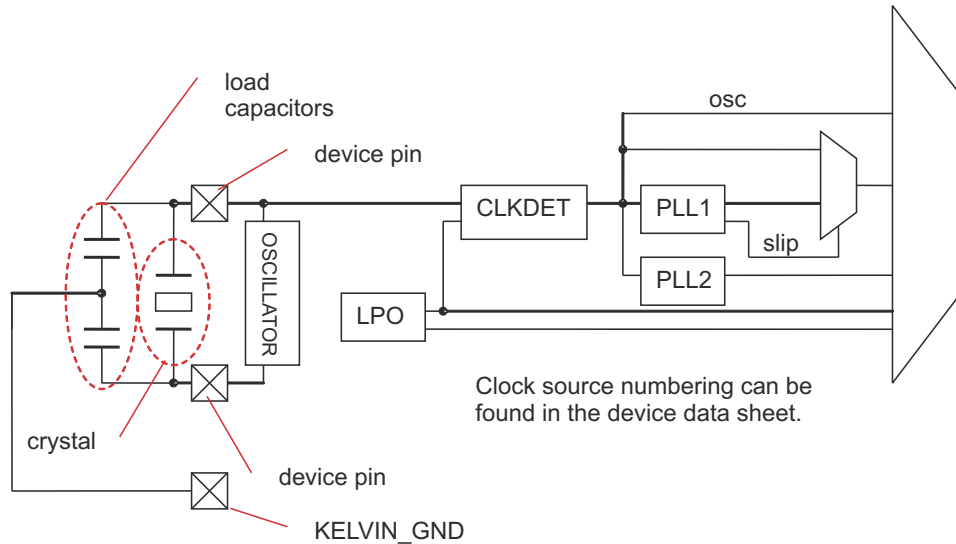
- The oscillator may drive a crystal/resonator or be driven from an external source
- The clock detect provides continuous monitoring of the oscillator frequency and provides an automatic switch over to a free-running clock in case of oscillator failure.
- The FM-PLL module can be operated in either modulation or non-modulation mode.
- The phase-frequency detector assures lock to the fundamental reference frequency.

- $$f_{PLL} = \frac{f_{OSCIN}}{NR} \frac{NF}{OD \times R}$$
 (2)
  - Configurable prescale divider (NR) for the input clock
  - Configurable multiplier (NF)
  - Configurable postscale dividers (OD, R)
- The PLL may be used with modulation enabled.
  - Configurable modulation frequency (NS)
  - Configurable modulation depth (NV)
- The slip control circuitry provides flexible response to a PLL failure (slip) including reset or automatic switch over to oscillator.

## 10.2 Quick Start

The purpose of this section is to provide an overview of how to configure the oscillator and PLL clock paths on power-up. More detailed descriptions are presented in later sections. [Figure 10-1](#) shows the oscillator and PLL clock paths.

**Figure 10-1. Clock Path From Oscillator Through PLL To Device**



While power-on reset is asserted (low), the oscillator and low power oscillator (LPO) are enabled and start-up by default. After power-on reset is released to a high level, the clock detect circuit (CLKDET) begins to monitor the oscillator. If the oscillator is within a valid range, the oscillator becomes the default clock for the device as it exits reset; if the oscillator is not within a valid range, the clock detect selects the high-frequency low power oscillator as the default clock for the device.

The low power oscillator has a wide frequency range which also creates a large valid window for the clock detect; in order to refine the clock detect window, the low power oscillator can be trimmed. The initial trim value is stored in one-time programmable section of the flash memory, address 0xF008\_01B4. Bits 31:16 of this word contain a 16 bit value that may be programmed into LPOMONCTL(15:0) in order to initialize the trim for both HF LPO and LF LPO. Software should read the initial trim values from flash and write them to the control register.

The PLL is disabled by default on power-up. The PLL control registers (PLLCTL1 and PLLCTL2) must be configured to set the desired output frequency. Then, the system PLL may be enabled (CSDISCLR.1). Similarly, the 2nd PLL must be configured in PLLCTL3 and enabled (CSDISCLR.6). Each PLL has a valid bit that indicates the PLL is locked (CSVSTAT.1 and CSVSTAT.6 respectively).

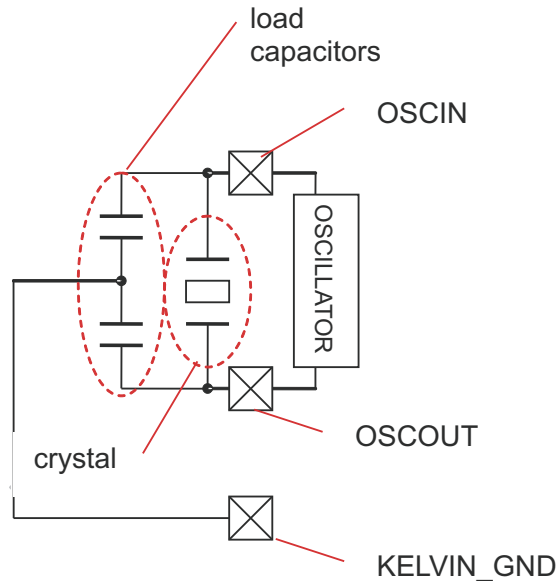
Prior to selecting the PLL clock as the source for a clock domain (e.g. GCLK, HCLK, ACLKV1), the domain and modules on the domain must be configured to accept the new frequency. An example of a module that should be configured prior to selecting the PLL as clock source for GCLK and HCLK is the memory wrapper to insure that access times are maintained correctly.



### 10.3 Oscillator

The clock generation path through the PLL begins with the oscillator. The oscillator consists of three separate pads -- OSCIN, OSCOUT, and Kelvin\_GND (see [Figure 10-2](#)).

**Figure 10-2. Clock Generation Path**



The oscillator is responsible for two independent functions:

1. The oscillator is responsible for generating positive feedback in the external crystal/resonator with appropriate load and tank circuitry. At start-up, the oscillator amplifies random noise. The external circuitry acts like a band-pass and selects the crystal/resonator frequency to provide as positive feedback into the amplifier. The positive feedback increases the amplitude of the output waveform into the crystal/resonator (and the load circuitry), and the voltage waveform shows an envelope of increasing amplitude. The oscillator can drive a crystal frequency that is within the data sheet range  $f_{(OSC\_Sine)}$ .

Looking at the input waveform into OSCIN, the voltage waveform is an AC-coupled, filtered version of the OSCOUT waveform. The band-pass functionality of the crystal/resonator removes any distortion that might exist on the OSCOUT waveform, leaving a sinusoidal input waveform.

---

**NOTE: Vendor validation of Resonators/Crystals**

The crystal is a very tight bandpass filter while a resonator is a somewhat wider bandpass. The load circuitry pulls the center frequency of the bandpass.

Texas Instruments strongly encourages each customer to submit samples of the device to the resonator/crystal vendor for validation. The vendor is equipped to determine what load capacitances will best tune their resonator/crystal to the microcontroller device for optimum start-up and operation over temperature and voltage extremes. The vendor also factors in margins for variations in the microcontroller process.

---

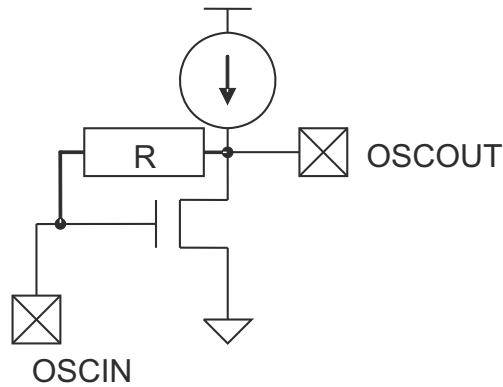
2. The oscillator is also responsible for squaring-up the input waveform. This squaring-up converts the sinusoid into a square wave at the core logic levels. The input path limits the input frequency range as a low-pass filter with a cutoff frequency at the data sheet frequency  $f_{[comparator]}$ .

The oscillator has a frequency range that is determined by the driving capability of external crystals/resonators (feedback path). If a clock is driven directly into the oscillator, then the feedback path is not relevant and the frequency range is determined solely by the forward path (which typically allows a higher frequency); the device can support inputs within the data sheet range  $f_{(OSC\_Sq)}$ .

### 10.3.1 Oscillator Implementation

The oscillator operates at 3.3V and uses a constant current source to drive current onto the OSCOUT node. An internal transistor shunts the current (and current from the external circuitry) to GND. This current steering drives the voltage waveform on OSCOUT.

Figure 10-3. Oscillator Implementation



### 10.3.2 Oscillator Enable

The oscillator is enabled asynchronously when nPORRST is low.

The oscillator is enabled by clearing bit 0 in CSDIS or setting bit 0 in CSDISCLR (in the System module). The bit sends a start signal to the oscillator. Bit 0 of CSDIS register is cleared to 0 by default on a system or power-on reset so that the oscillator starts-up by default. After the oscillator swings at a high-enough amplitude to pass an input clock into the core domain and nPORRST is released, 1024 oscillator periods are counted before setting the OSCILLATOR VALID flag in the System module. The oscillator generates clock source 0 in the global clock module (GCM).

### 10.3.3 Oscillator Disable

Clock sources (e.g. OSC, PLL) can be disabled by writing the CSDIS or CSDISSET registers within the system module. These bits *allow* the clock source to disable but do not force the behavior until the clock is no longer used as the source for a clock domain (e.g. GCLK, VCLK, VCLK2, RTICLK). The Oscillator VALID flag is cleared after clock disable is asserted (which occurs after all clock domains are stopped).

The oscillator disable signal places the oscillator into a low power state, disconnects the feedback (bias) resistor between OSCIN and OSCOUT, and OSCIN is grounded.

## 10.4 Low Power Oscillator and Clock Detect (LPOCLKDET)

The Low Power Oscillator (LPO) is comprised of two oscillators -- HF LPO and LF LPO -- in a single macro. The low power oscillator and clock detect (LPOCLKDET) uses a relaxation oscillator to generate an internal clock whose frequency is NOT tightly controlled. This frequency is used to monitor the oscillator input frequency and is also available as an independent clock source in the GCM.

The LPO produces two frequencies:

- high-frequency low power oscillator (HF LPO) with a nominal frequency of 10MHz and a range from 5.6MHz to 22.6MHz; the HF LPO generates clock source 5 in the GCM.
- low frequency low power oscillator (LF LPO) with a nominal frequency of 105KHz and a range from 55KHz to 200KHz; the LF LPO generates clock source 4 in the GCM.

A single current source drives current onto a capacitor; when the voltage on the capacitor exceeds some threshold, the clock toggles. The LPO uses a single current source and the two different comparators to generate the HF LPO and LF LPO frequencies. The LPO is controlled by 4 different bit fields -- CSDIS.(5:4), HFTRIM(4:0), LFTRIM(4:0), and BIASEN.

- CSDIS.5 enables/disables the comparator that generates HF LPO
- CSDIS.4 enables/disables the comparator that generates LF LPO
- the HF TRIM and LF TRIM bit fields vary the current into the comparator to independently trim the HF LPO and LF LPO frequencies (see Section 7.6.9)
- BIAS ENABLE (LPOMONCTL.24) enables/disables the current source which drives the LPO

### 10.4.1 Clock Detect

The LPO HF clock frequency is typically near 10MHz but ranges from 5.6MHz to 22.6MHz. The clock detect establishes a window for the oscillator by:

$OSCIN > HF\ LPO_{min} / 4$	$OSCIN / 4 < HF\ LPO_{max}$
$OSCIN > 5.6[MHz] / 4 = 1.4[MHz]$	$OSCIN < 4 * 22.6 = 90.4[MHz]$

The clock detect circuit works by checking for a rising edge on one clock (oscillator or HF LPO) between rising edges of the other clock. The result is that in addition to flagging incorrect, repeating frequencies, the circuit also fails due to transient conditions.

---

**NOTE: Clock Detection of Oscillator MUST be Disabled Before Disabling HF LPO**

The HF LPO frequency is the comparison frequency for the oscillator. The clock detection must be disabled prior to disabling the HF LPO frequency.

If the clock detection is NOT disabled prior to disabling the HF LPO, the clock detect circuitry will fail the oscillator as too fast (compared to the non-existent HF LPO). The clock detect circuitry will switch to the non-existent clock, leaving the device without a valid clock.

---

### 10.4.2 Behavior on Oscillator Failure

If the oscillator frequency fails, the clock detects supplies:

- the HF LPO clock to GCM clock source 0 instead of the oscillator
- the HF LPO clock to GCM clock source 1 instead of the PLL

The HF LPO signal will be available as three different clock sources:

- GCM clock source 0 (replacing the oscillator)
- GCM clock source 1 (replacing the PLL)
- GCM clock source 4 as HF LPO

The automatic switch-over from oscillator to HF LPO allows the application to execute at a reduced frequency and respond to a problem with the external crystal/resonator. During and after an oscillator failure, the oscillator VALID bit is set along with the OSC FAIL flag.

It is useful to explicitly change the GHVSRG register, defining the current clock source for GCLK/HCLK/VCLK domains, to the HF LPO after an oscillator failure.

When reset on oscillator failure is set, PLLCTL1.23 (ROF), the device responds to an oscillator failure by generating a device reset.

### 10.4.3 Recovery from Oscillator Failure

If the oscillator fails, the clock detect switches the HF LPO frequency onto the oscillator source into the GCM. The OSCFAIL flag is also set within the System module.

The oscillator may be re-enabled (though if the failure was caused by a hard-fault, the re-enable will fail) through the following procedure:

1. Switch all clock domains from the oscillator to the HF LPO (e.g. GHVSRG uses HF LPO, ACLKV uses HF LPO or VCLK,...).
2. If the PLL is used, disable the PLL with CSDISSET.
3. Disable the oscillator with CSDISSET. This action resets the clock detect and allows the oscillator to propagate through GCM clock source 0.
4. Re-enable the oscillator with CSDISCLR.
5. Clear the OSC FAIL flag in the System register by writing a 1 to the bit. The PLL slip bits may also be set on an oscillator failure. These can also be cleared.
6. Switch the clock domains back to the oscillator.
7. Re-enable the PLL.

---

**NOTE: Clock Re-Enable Procedure Will Fail If Caused by a Hard Failure**

Although it is possible to re-enable the oscillator after a failure, if the oscillator failure was caused by a hard fault (e.g. disconnected crystal/resonator terminal), the re-enable process will fail.

---

### 10.4.4 LPOCLKDET Enable

The LPO is enabled by default while nPORRST is low. During this time, the current source initializes, holding the relaxation oscillator in reset until initialized. After the current source releases the HF LPO and the LF LPO, these clock frequencies slew to their final frequencies; the final frequency may be achieved while nPORRST is active or after its release. After, nPORRST is released, the HF LPO Valid signal is set 32 HF LPO clock cycles later.

The clock detect is enabled once the oscillator and HF LPO are valid. Because an oscillator failure could occur from reset, the clock detect logic must provide an override path. If the HF LPO is valid and the oscillator is not valid, the clock detect circuitry will become active (overriding the oscillator invalid signal) after 16K LF LPO cycles (about 200ms).

### 10.4.5 LPOCLKDET Disable

#### 10.4.5.1 Disable Clock Detect

It is possible to disable the clock detect circuitry. For protection, this clock detect disable employs a 2-bit key:

- RANGE DET ENA SSET (CLKTEST.24) must be set to 1
- RANGE DET CTRL (CLKTEST.25) must be cleared to 0

In this case, the LPO HF and LF clocks are still active but the clock detect circuitry is disabled. The clock detect unconditionally switches GCM\_CLK\_SRC(0) back to the oscillator so care should be taken to insure that the oscillator is good before disabling the clock detect circuitry.

### 10.4.5.2 Disable LPO HF and LF clocks

The LPO may be disabled by holding the relaxation oscillator clocks (HF and LF) in reset. The clock detect must be disabled, and any clock domains using either HF or LF clocks must be switched to a different clock source. The LPO HF clock is reset by setting CSDIS.5; CSDISSET.5 is an easy way to set specific bits without disturbing the rest of the register. The HF LPO clock disables several HF LPO cycles after CSDIS is set.

Similarly, the LPO LF clock is reset by setting CSDIS.4, and in a similar way CSDISSET.4 can set the specific CSDIS register bit without using a read-modify-write construction. The LF LPO disables several LF LPO cycles after CSDIS is set.

Restarting the LPO clocks from this condition is fast and is known as a warm re-start. The CSDISCLR register allows the user to clear CSDIS bits without using a read-modify-write code-construct.

### 10.4.5.3 Disable LPO Current Bias

The LPO current source may be disabled after the clock detect is disabled and HF and LF clock sources are disabled. Turning off this current source places the LPOCLKDET into its lowest power configuration. The bias may be disabled by clearing the BIAS ENABLE bit (LPOMONCTL.24).

Restarting the LPO when the bias current has been disabled requires the current source to initialize first and is, therefore slower than a warm re-start; re-enabling the LPO from this condition is known as a warm re-start (similar to what happens during nPORRST active).

## 10.4.6 Trimming the HF LPO Oscillator

The HF LPO range varies considerably around 10MHz from device to device. In order to provide tighter monitoring of the crystal/resonator, it is useful to trim the oscillator. During device test, a trim value is written into the one-time programmable section of the flash memory (OTP), address 0xF008\_01B4. Bits 31:16 of this OTP word contain a 16 bit value that may be programmed into LPOMONCTL(15:0) in order to initialize the trim for both HF LPO and LF LPO.

When trimming the HF LPO, it is recommended to step the trim value so as not to make a large change to any TRIM setting.

After the initial trim, further trimming may be done in LPOMONCTL, using the dual clock compare module (DCC, please see Dual Clock Compare User's Guide) in order to determine the resultant frequency. This module allows for comparison of two clock frequencies. Once the HF LPO is determined to be in-range with the initial HFTRIM setting from the OTP, the crystal oscillator may be used as a reference against which the HF LPO and LF LPO may be further adjusted.

### 10.5 PLL

The following bit fields from PLLCTL1 and PLLCTL2 configure the PLL:

- REFCLKDIV[5:0]
- PLLMUL[15:0]
- ODPLL[2:0]
- PLLDIV[4:0]
- SPR\_AMOUNT[8:0]
- SPREADINGRATE[8:0]
- FMENA

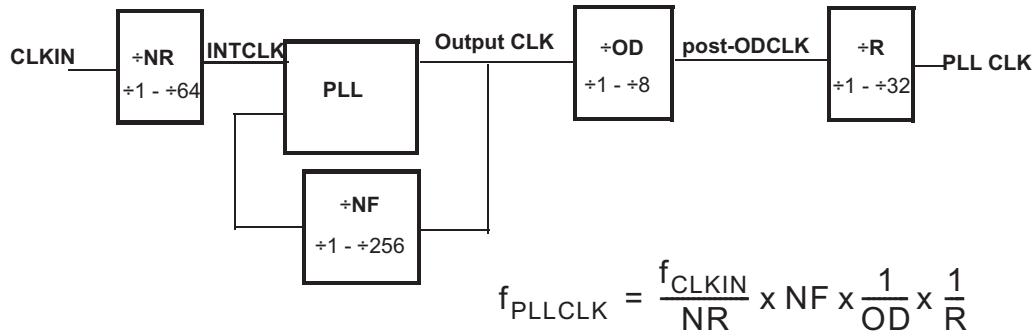
The PLL is responsible for synthesizing an output frequency from the input clock (from the oscillator); [Figure 10-4](#) shows a simple block diagram of the PLL. The FM-PLL divides the reference input for a lower frequency input into the PLL ( $f_{INTCLK} = f_{CLKIN}/NR$ ). The PLL multiplies this internal frequency by NF to get the VCO output clock frequency ( $f_{Output\ CLK} = f_{INTCLK} * NF$ ). The PLL 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. This output clock, PLL CLK, sources GCM clock source 1. Valid frequencies are shown in [Table 10-1](#) while [Table 10-2](#) shows how that encoding is generated from the PLL bit fields.

[ $f_{(post\_ODCLK)}$  and  $f_{(HCLK)}$  are data sheet parameters.]

**Table 10-1. Valid Frequency Ranges for PLL**

	Frequency Limit
$f_{CLKIN}$	$f_{(OSC\_Sqr)}$
$f_{INTCLK}$	1MHz - $f_{(OSC\_Sqr)}$
$f_{Output\ CLK}$	150MHz - 550MHz
$f_{post-ODCLK}$	$f_{(post\_ODCLK)}$
$f_{PLL\ CLK}$	$f_{(HCLK)}$

**Figure 10-4. Operation of the FM-PLL module**



**Table 10-2. PLL Value Encoding**

	PLL
NR	$NR = REFCLKDIV[5...0] + 1$ (3)

**Table 10-2. PLL Value Encoding (continued)**

	PLL	
	Non-modulated: $NF = \frac{(PLLMUL[15...0] + 1 \ll 8)}{256}$	(4)
NF	Modulated: $NF = \frac{(PLLMUL[15...0] + MULMOD[8...0] + 1 \ll 8)}{256}$	(5)
NV	$NV = \frac{(SPRAMOUNT[8...0] + 1)}{2048}$	(6)
NS	$NS = SPRRATE[8...0] + 1$	(7)
OD	$OD = ODPLL[2...0] + 1$	(8)

**NOTE: ODPLL change should occur prior to enabling asynchronous clock domains**

Since changing the ODPLL bit-field causes the PLL CLK to be gated, these changes to ODPLL should be completed before configuring a clock domain for an asynchronous clock source. Some clock domains (e.g. RTICKL, AVCLK) require a frequency relationship to the VCLK.

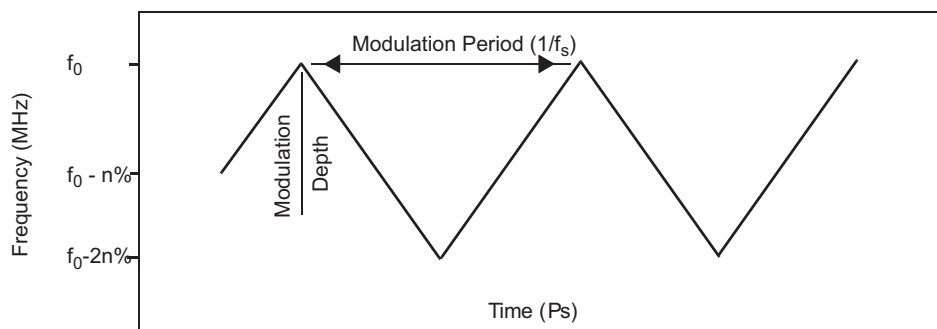
$$f_{VCLK} \geq 3 \times \frac{f_{RTISRC}}{RTIDIV}$$

If the PLL is clocking VCLK and it is stopped for some cycles, then the frequency relationship is temporarily violated.

Many asynchronous domains require frequency relationships between VCLK and the asynchronous domain. Therefore, if the PLL clock is the source for GCLK, HCLK and VCLK, then the gating produces a short-term change in the PLL clock frequency (and hence also the VCLK frequency). As such, this frequency change could violate the requirements for an asynchronous clock domain.

**10.5.1 Modulation**

Optionally, the frequency can be modulated, that is, a controlled jitter is introduced onto the baseline frequency of the PLL. This modulation mechanism is not shown in Figure 10-4. When the PLL is used in the modulating mode, the programmable modulation block varies the PLL frequency from the baseline frequency ( $f_{\text{baseline}} = (f_{\text{CLKIN}}/NR) * NF / (OD * R)$ ) to  $f_{\text{baseline}} * (1 - 2 * \text{Depth})$  in a period defined by  $1/f_s$ ; the modulation waveform is triangular and should be enabled after lock.



The modulation is digital and the spreading profile is triangular, down-spread which implies:

- the modulation waveform is composed of a series of frequency steps
- the modulation frequency and modulation depth are both well controlled due to their digital character
- the average frequency during modulation is lower than the average frequency prior to enabling modulation. The depth of modulation, however, sets the new average frequency.
- the modulation frequency must be selected slower than the loop bandwidth. From a practical perspective, NS should be near 20.

The modulation fields have a simple geometric meaning:

- the modulation step size is

$$\frac{NV}{NF} \times f_{OutputCLK} \quad (9)$$

- the number of steps per modulation period is 2\*NS
- the modulation depth is given by

$$\frac{NS}{2} \times \frac{NV}{NF} \times f_{OutputCLK} \quad (10)$$

- the modulation frequency is

$$T_{mod} = \frac{f_{osc}}{2 \times NR \times NS} \quad (11)$$

---

**NOTE: Modulation should be enabled after Lock**

Enable modulation after the lock is completed.

---

## 10.5.2 PLL Output Control

The outputs from the PLL are the output clock, slip signals and VALID.

- RFSLIP -- the RFSLIP signal indicates that the Output CLK is running *too fast* relative to INTCLK and sets a status flag in the System's GLBSTAT register if the slip signal is active during normal PLL operation; the RFSLIP bit is masked off while the PLL is not active and during the PLL's lock period.
- FBSLIP -- the FBSLIP signal indicates that the Output CLK is running *too slow* relative to INTCLK and sets a status flag in the System's GLBSTAT register if the slip signal is active during normal PLL operation; the FBSLIP bit is masked off while the PLL is not active and during the PLL's lock period.
- PLL Slip -- OR of the two PLL slip signals. Typically this signal is used to generate a consolidated slip signal to the device (e.g. error logic or exception generation). Also used to gate VALID.

---

**NOTE: Clearing Slip Bits**

In order to clear any of the slip bits, it is necessary to disable the PLL first.

---

- VALID -- is driven based upon whether the output clock -- PLL CLK -- is gated or not. However, the VALID signal is dependent upon the PLL Slip signals so that VALID cannot be set if either slip signal is active.
- PLL Clock -- The PLL output clock runs at the programmed frequency. When enabled, it takes some time to acquire the programmed frequency (see [Section 10.5.2.1](#)). Similarly, the disable has some timing/constraints (see [Section 10.5.2.2](#)).

### 10.5.2.1 PLL Enable

After setting the PLL control registers, the clock source is enabled by clearing the appropriate bit in CSDIS or setting the appropriate bit in CSDISCLR (in the System module). The bit sends a signal to the PLL which starts the process of enabling the PLL.

1. The PLL checks to make sure that the oscillator is ON. If not, it turns the oscillator ON.



2. The PLL begins a locking process in which the PLL slews from a starting frequency point to the programmed frequency. During this lock period, the PLL slip signals are typically active, and the PLL masks off the signals during this phase. The lock phase takes the following length of time:

	<b>LPOMONCTRL.16 = 0 (OSCIN &lt;= 20MHz) Counters clocked by OSCIN /1</b>	<b>LPOMONCTRL.16 = 1 (OSCIN &gt; 20MHz) Counters clocked by OSCIN /4</b>
Lock	$T_{\text{Lock}} = (127 \times T_{\text{OSCIN}}) + (1024 \times \text{NR} \times T_{\text{OSCIN}})$	$T_{\text{Lock}} = (127 \times 4 \times T_{\text{OSCIN}}) + (1024 \times \text{NR} \times T_{\text{OSCIN}})$
Enable clocks after lock	$T_{\text{Enable}} = 6 \times T_{\text{OSCIN}}$	$T_{\text{Enable}} = 18 \times T_{\text{OSCIN}}$

3. After the lock phase is complete (when lock counters expire), the PLL releases the slip signals to the system.
4. Then, after the slip signals are released and a delay to enable the clocks, the clock is released to the system and the VALID bit for the PLL is set in the Clock Source Valid register (CSVSTAT).

### 10.5.2.2 PLL Disable

Clock sources (e.g. OSC, PLL) can be disabled by writing the CSDIS or CSDISSET registers within the system module. These bit *allow* the clock to disable but do not force the behavior until the clock is no longer used as the source for a clock domain (e.g. GCLK, VCLK, VCLK2, RTICKL).

The PLL receives a signal to disable after the clock is no longer used by any clock domain. Within the PLL, the clock is disabled and the VALID bit becomes inactive. Then the PLL is placed into a low power state after the following length of time:

	<b>LPOMONCTRL.16 = 0 Counters clocked by OSCIN /1</b>	<b>LPOMONCTRL.16 = 1 Counters clocked by OSCIN /4</b>
Disable clocks after lock	$T_{\text{Enable}} = 150 \times T_{\text{OSCIN}}$	$T_{\text{Enable}} = 582 \times T_{\text{OSCIN}}$

### 10.5.2.3 OD-divider Change

The PLL gates the clock if the ODPLL bit-field is changed while the PLL is active. The output clock from the PLL is gated for 3 or 12 OSCIN clock cycles. As the post-ODCLK is gated in the low phase, the output clock to the device -- PLL CLK -- may be gated in a high or low phase though the transition is always glitchless.

	<b>LPOMONCTRL.16 = 0 Counters clocked by OSCIN /1</b>	<b>LPOMONCTRL.16 = 1 Counters clocked by OSCIN /4</b>
Change ODPLL	$T_{\text{ODPLL}} = 3 \times T_{\text{OSCIN}}$	$T_{\text{ODPLL}} = 3 \times 4 \times T_{\text{OSCIN}}$

**NOTE: ODPLL change should occur prior to enabling asynchronous clock domains**

Since changing the ODPLL bit-field causes the PLL CLK to be gated, these changes to ODPLL should be completed before configuring a clock domain for an asynchronous clock source. Some clock domains (e.g. RTICKL, AVCLK) require a frequency relationship to the VCLK.

$$f_{\text{VCLK}} \geq 3 \times \frac{f_{\text{RTISRC}}}{\text{RTIDIV}}$$

If the PLL is clocking VCLK and it is stopped for some cycles, then the frequency relationship is temporarily violated.

Many asynchronous domains require frequency relationships between VCLK and the asynchronous domain. Therefore, if the PLL clock is the source for GCLK, HCLK and VCLK, then the gating produces a short-term change in the PLL clock frequency (and hence also the VCLK frequency). As such, this frequency change could violate the requirements for an asynchronous clock domain.

### 10.5.2.4 Changing the PLL Operating Point While the PLL is Active

Once the valid bit is set, software may change values to the PLL. If the change of values results in a small percentage change to the VCO frequency ( $\Delta f_{\text{OutputCLK}} < 0.1 \times f_{\text{OutputCLK}}$ ), then these changes can be done on-the-fly. In this mode, the values are updated into the PLL synchronously, and the PLL re-locks to the new value without gating the clocks or the slip bits. If the operating point change is too large, then the slip bits will be set.

Conversely, if the changes to the VCO frequency are large, then the PLL should be disabled prior to changing the values. Typically, any change to the REFCLKDIV field or large changes to the PLLMUL field requires a complete disable-and-relock strategy.

### 10.5.2.5 Summary of PLL Timings

In addition to controlling the lock period and disabling the clock during an ODPLL change, the PLL also generates reset delays. When power-on reset is released (nPORRST 0 --> 1), that release is delayed by 1024 OSCIN cycles so that it is released at the same time that the oscillator valid is asserted. The system reset release is delayed by an additional 8 oscillator clock cycles.

**Table 10-3. State Machine Timings**

	<b>LPMONCTRL.16 = 0 Counters clocked by OSCIN /1</b>	<b>LPMONCTRL.16 = 1 Counters clocked by OSCIN /4</b>
nPORRST delay	$T_{\text{nPORRST}} = 1024 \times T_{\text{OSCIN}}$	$T_{\text{nPORRST}} = 1024 \times T_{\text{OSCIN}}$
nRST delay	$T_{\text{nRST}} = 1032 \times T_{\text{OSCIN}}$	$T_{\text{nRST}} = 1032 \times T_{\text{OSCIN}}$
OSC valid	$T_{\text{OSCVLID}} = 1024 \times T_{\text{OSCIN}}$	$T_{\text{OSCVLID}} = 1024 \times T_{\text{OSCIN}}$
Lock	$T_{\text{Lock}} = (127 \times T_{\text{OSCIN}}) + (1024 \times \text{NR} \times T_{\text{OSCIN}})$	$T_{\text{Lock}} = (127 \times 4 \times T_{\text{OSCIN}}) + (1024 \times \text{NR} \times T_{\text{OSCIN}})$
Enable clocks after lock	$T_{\text{Enable}} = 6 \times T_{\text{OSCIN}}$	$T_{\text{Enable}} = 18 \times T_{\text{OSCIN}}$
Disable clocks after lock	$T_{\text{Enable}} = 150 \times T_{\text{OSCIN}}$	$T_{\text{Enable}} = 582 \times T_{\text{OSCIN}}$
Change ODPLL	$T_{\text{ODPLL}} = 3 \times T_{\text{OSCIN}}$	$T_{\text{ODPLL}} = 3 \times 4 \times T_{\text{OSCIN}}$

### 10.5.3 Behavior on PLL Fail

The PLL allows flexible response to a PLL failure (slip). Like the oscillator, the PLL clock is configured by default to automatically switch-over to the oscillator in case of a PLL slip. (In this case, the oscillator sources GCM clock source 1 as well as GCM clock source 0. Also, if the oscillator fails, LPO HF is sourced to both GCM clock sources 0 and 1.)

The PLL slip outputs indicate that the PLL is running either too fast or too slow. These error output toggle when the PLL is locking and when the PLL is disabling. The PLL blocks these slip outputs during these times, leaving them active only while the PLL is active.

A slip after the PLL has locked and while it is active is an indication of a PLL failure. The PLL provides slip-filtering which enhances the flexibility of the PLL's response to failure. The slip-filtering circuit samples the slip based on HF LPO. The filter defines the number of consecutive HF LPO cycles for which the slip signal must be active before the slip is recognized. This slip is latched in the System status bit RFSLIP and FBSLIP.

The PLL may enable/disable the automatic switch over as well as the error signaling; if the error signaling is enabled, a PLL slip may be configured to generate a reset. The automatic switch-over and suppression of the error signals are controlled by the bypass on slip bit field -- BPOS[1:0] (PLLCTL1.(30:29)). When BPOS[1:0] is disabled (BPOS[1:0] = 10b):

- automatic response to the PLL slip is prevented
- ESM/exception is NOT generated
- reset on slip is not generated regardless of the state of the ROS bit
- status bits are set on a PLL slip independent of BPOS[1:0]

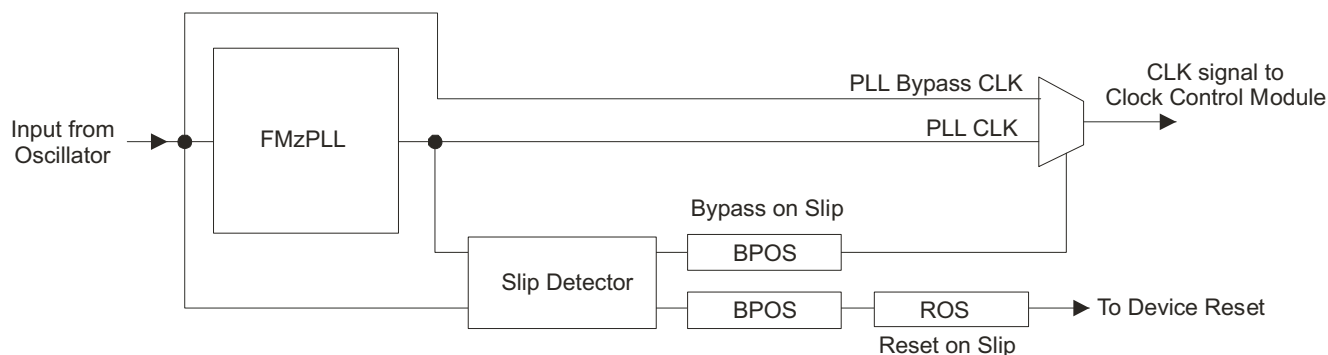
When BPOS[1:0] is enabled (BPOS[1:0] = 00b OR 01b OR 11b):

- PLL slip causes the clock source into GCM clock source 1 to shift from the PLL to the oscillator

- ESM/exception is generated
- reset on slip is generated if ROS is set

The effect of BPOS[1:0] on the system is shown in [Figure 10-5](#).

**Figure 10-5. PLL Slip Detection & Reset/Bypass Block Diagram**



### 10.5.4 Recovery From a PLL Failure

If the PLL fails, PLL slip causes the clock source into GCM clock source 1 to shift from the PLL to the oscillator. The RFSLIP or FBSLIP flag is also set within the System module.

The PLL may be re-enabled (though if the failure was caused by a hard-fault, the re-enable will fail) through the following procedure:

1. Switch all clock domains from the PLL to the oscillator (e.g. GHVSRV uses oscillator, ACLKV uses oscillator or VCLK,...).
2. Disable the PLL with CSDISSET. This action disables the PLL and causes the slip signal to no longer be driven. Valid is not released until the slip is cleared.
3. Clear the RFSLIP or FBSLIP flag in the System register by writing a 1 to the bit.
4. Re-enable the PLL with CSDISCLR.
5. Switch the clock domains back to the PLL.

### 10.5.5 PLL Modulation Depth Measurement

The PLL contains a circuit for estimating the depth of the modulation. The circuit counts clock edges over a fixed window of the modulation waveform (SSW\_CAPTURE\_COUNT in SSWPLL2) and clock edges over the entire waveform (SSW\_CLKOUT\_COUNT in SSWPLL3). The capture ends after a pre-determined number of clock edges in SSW\_CLKOUT\_COUNTER as set in TAP\_COUNTER\_DIS. There are  $2 \cdot NR$  windows per modulation waveform. The procedure for estimating the modulation depth is:

1. While GCLK is sourced by the oscillator and the PLL is enabled with modulation, configure SSWPLL1 as follows:
  - (a) CAPTURE\_WINDOW\_INDEX is set equal to NR
  - (b) COUNTER\_RESET is set
  - (c) TAP\_COUNTER\_DIS is set to disable the measurement after SSW\_CLKOUT\_COUNT captures this number of clocks. The measurement is disabled after the set tap is set AND the modulation cycle ends.
  - (d) Ensure that EXT\_COUNTER\_EN is cleared.
2. Ensure that both SSW\_CAPTURE\_COUNT and SSW\_CLKOUT\_COUNT are cleared (by the COUNTER\_RESET).
3. Set COUNTER\_EN and clear COUNTER\_RESET. This step releases the reset and enables the counter to begin counting.
4. After a wait loop, poll for COUNTER\_READ\_READY to set. After the bit is set, read SSW\_CAPTURE\_COUNT and SSW\_CLKOUT\_COUNT.

5. Compute the modulation depth as:

$$Depth = abs\left(1 - \frac{2 \times NR \times SSW\_CAPTURE\_COUNT}{SSW\_CLKOUT\_COUNT}\right) \quad (12)$$

### 10.5.6 PLL Frequency Measurement Circuit

The same circuit that is used to measure modulation depth is also available to measure the average frequency of the PLL. In this mode, the PLL output (before the R-divider) is captured in SSW\_CLKOUT\_COUNT while the oscillator is captured in SSW\_CAPTURE\_COUNT. The procedure for using the PLL frequency measurement circuit is:

1. While the PLL is enabled, set EXT\_COUNTER\_EN.
2. Set COUNTER\_EN. This bit clears both SSW\_CAPTURE\_COUNT and SSW\_CLKOUT\_COUNT and then immediately enables for counting.
3. Wait for some software delay loop.
4. Clear COUNTER\_EN. Wait for COUNTER\_READ\_READY to set. Read both SSW\_CAPTURE\_COUNT and SSW\_CLKOUT\_COUNT and compute the ratio of PLL multiplication as:

$$\frac{NF}{NR \times OD} = \frac{SSW\_CAPTURE\_COUNT}{SSW\_CLKOUT\_COUNT} \quad (13)$$

5. Note that CAPTURE\_WINDOW\_INDEX, COUNTER\_RESET, TAP\_COUNTER\_DIS are not used in this procedure

### 10.5.7 PLL2

PLL2 drives GCM clock source 6.

The PLL is identical to PLL1, except modulation is disabled on this instance of the PLL. Also, the PLL typically does not clock the system, there is no automatic switch over feature. Any PLL error can be handled by the CPU.

## 10.6 Control Registers

The 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 FM-PLL is off at power-on. It may be turned on by clearing the CLK\_SRC1\_OFF bit in the CSDIS register of the System module. [CSDISCLR and CSDISSET also enable/disable the PLL and oscillator (and other clock sources).]

The LPOCLKDET module generates the OSC\_FAIL flag (located in GLBSTAT register) if a problem with the reference oscillator is detected. The slip signals are also registered in the GLBSTAT register in order to indicate the source of a clock failure.

The VALID bits for the clock sources are located in the CSVSTAT register within the System module.

The following sections describe the two PLL registers used in the system module. These registers support 8, 16, and 32-bit write accesses. The reset values for these registers are configured so that an input frequency in the range from 5MHz - 20MHz generates a valid clock.

**Table 10-4. PLL Module Registers**

Offset	Acronym	Register Description	Section
FFFF FF70h	PLLCTL1	PLL Control 1 Register	<a href="#">Section 10.6.1</a>
FFFF FF74h	PLLCTL2	PLL Control 2 Register	<a href="#">Section 10.6.2</a>
FFFF E100h	PLLCTL3	PLL Control 3 Register	<a href="#">Section 10.6.3</a>
FFFF E170h	CLKSLIP	PLL Clock Slip Control Register	<a href="#">Section 10.6.4</a>
FFFF FFA0h	GPREG1	General Purpose Register	<a href="#">Section 10.6.5</a>
FFFF FF24h	SSWPLL1	PLL Modulation Depth Measurement Control Register	<a href="#">Section 10.6.6</a>
FFFF FF28h	SSWPLL2	SSW PLL BIST Control Register 2	<a href="#">Section 10.6.7</a>
FFFF FF2Ch	SSWPLL3	SSW PLL BIST Control Register 3	<a href="#">Section 10.6.8</a>

**Table 10-5. LPOCLKDET Module Registers**

Offset	Acronym	Register Description	Section
FFFF FF88h	LPOMONCTL	LPO/Clock Monitor Control Register	<a href="#">Section 10.6.9</a>
FFFF FF8Ch	CLKTEST	Clock Test Register	<a href="#">Section 10.6.10</a>

### 10.6.1 PLL Control 1 Register (PLLCTL1)

Figure 10-6 illustrates this register and Table 10-6 provides the bit descriptions.

**Figure 10-6. PLL Control 1 Register (PLLCTL1) [offset = FFFF FF70h]**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
ROS	BPOS	PLLDIV						ROF	Reserv ed	REFCLKDIV						
R/W-0	R/W-1	R/W-1Fh						R/W-0	R-0	R/W-3h						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
PLLMUL																
R/W-5700h																

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

**Table 10-6. PLL Control 1 Register (PLLCTL1) Field Descriptions**

Bit	Field	Value	Description
31	ROS	0 1	Reset on PLL Slip User and privilege mode (read): Privilege mode (write): Do not reset system when slip is detected Reset when slip is detected <b>Note: BPOS (PLLCTL1.29:30) must be enabled in order to use Reset On Slip (ROS) functionality.</b>
30-29	BPOS	2h Other	Bypass of PLL Slip User and privilege mode (read): Privilege mode (write): Bypass on PLL Slip is disabled. If a PLL Slip is detected no action is taken. 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. <b>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.</b>
28-24	PLLDIV	0 1h : 1Fh	PLL Output Clock Divider $R = PLLDIV + 1$ $f_{PLL\ CLK} = f_{post-ODCLK} / R$ User and privilege mode (read): Privilege mode (write): $f_{PLL\ CLK} = f_{post-ODCLK} / 1$ $f_{PLL\ CLK} = f_{post-ODCLK} / 2$ continues in sequence $f_{PLL\ CLK} = f_{post-ODCLK} / 32$
23	ROF	0 1	Reset on Oscillator Fail User and privilege mode (read): Privilege mode (write): Do not reset system when oscillator is out of range. The ROF bit enables the OSC_FAIL condition to generate a system reset. If the ROF bit in the PLLCTL1 register is set when the oscillator fails, then a system reset occurs.
22	Reserved	0	Read returns 0. Writes have no effect.

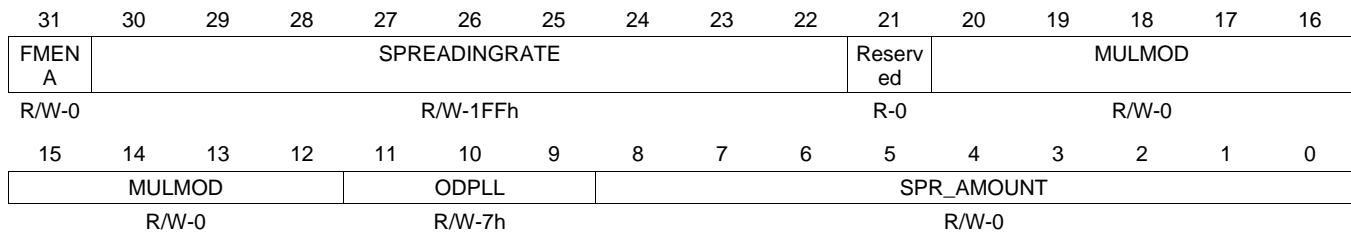
**Table 10-6. PLL Control 1 Register (PLLCTL1) Field Descriptions (continued)**

Bit	Field	Value	Description
21-16	REFCLK DIV	0 1h : 3Eh	<p>Reference Clock Divider</p> <p><math>NR = REFCLKDIV + 1</math> (62 -- max)</p> <p><math>f_{INT\ CLK} = f_{OSCIN} / NR</math></p> <p>User and privilege mode (read): Privilege mode (write):</p> <p><math>f_{INT\ CLK} = f_{OSCIN} / 1</math></p> <p><math>f_{INT\ CLK} = f_{OSCIN} / 2</math></p> <p>continues in sequence</p> <p><math>f_{INT\ CLK} = f_{OSCIN} / 63</math></p> <p><b>Note: Typically, this value should not be changed while the PLL is active.</b></p>
15-0	PLLMUL	100h 180h 5C00h : FF00h	<p>PLL Multiplication Factor</p> <p>Valid multiplication factors are from 1 to 256.</p> <p>For non-modulated mode: <math>NF = (PLLMUL / 256) + 1</math></p> <p>For modulated mode, MULMOD bit field is added as in <a href="#">Table 10-2</a></p> <p><math>f_{VCO\ CLK} = f_{INT\ CLK} \times NF</math></p> <p>User and privilege mode (read): Privilege mode (write):</p> <p><math>f_{VCO\ CLK} = f_{INT\ CLK} \times 1</math></p> <p><math>f_{VCO\ CLK} = f_{INT\ CLK} \times 1.5</math></p> <p><math>f_{VCO\ CLK} = f_{INT\ CLK} \times 93</math></p> <p>continues in sequence</p> <p><math>f_{VCO\ CLK} = f_{INT\ CLK} \times 256</math></p>

### 10.6.2 PLL Control 2 Register (PLLCTL2)

The PLLCTL2 register controls the frequency modulated mode of operation of the FM-PLL and the OD post-divider. The frequency modulation option is available for applications that have critical EMC considerations. [Figure 10-7](#) illustrates this register and [Table 10-7](#) provides the bit descriptions.

**Figure 10-7. PLL Control 2 Register (PLLCTL2) [offset = FFFF FF74h]**



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

**Table 10-7. PLL Control 2 Register (PLLCTL2) Field Descriptions**

Bit	Field	Value	Description
31	FMENA	0 1	Frequency Modulation Enable User and privileged mode (read): Privileged mode (write): 0 Disable frequency modulation 1 Enables frequency modulation
30-22	SPREAD INGRAT E	0 1h : 1FFh	$NS = SPREADINGRATE + 1$ $f_{mod} = f_s = f_{INT\ CLK} / (2 * NS)$ User and privileged mode (read): Privileged mode (write): 0 $f_{mod} = f_s = f_{INT\ CLK} / (2 * 1)$ 1h $f_{mod} = f_s = f_{INT\ CLK} / (2 * 2)$ : continues in sequence 1FFh $f_{mod} = f_s = f_{INT\ CLK} / (2 * 512)$
21	Reserved	0	Read returns 0. Writes have no effect.
20-12	MULMO D	0 8h 9h FFh 1FFh	Multiplier Correction when Modulation Enabled $MUL\_when\_MOD = 0$ when FMENA = 0 $MUL\_when\_MOD = (MULMOD / 256)$ when FMENA = 1 Incremental adder to CLKF (and NF) value when the modulation is enabled. When modulation is disabled, the value in this register is ignored. User and privileged mode (read): Privileged mode (write): 0 No adder to CLKF or NF when FMENA = 1 8h $MUL\_when\_MOD = 8/256$ 9h $MUL\_when\_MOD = 9/256$ FFh $MUL\_when\_MOD = 255/256$ 1FFh $MUL\_when\_MOD = 511/256$
11-9	ODPLL	0 1h : 7h	Internal PLL Output Divider User and privileged mode (read): Privileged mode (write): $OD = ODPLL + 1$ $f_{post-ODCLK} = f_{VCO\ CLK} / OD$ 0 $f_{post-ODCLK} = f_{VCO\ CLK} / 1$ 1h $f_{post-ODCLK} = f_{VCO\ CLK} / 2$ : continues in sequence 7h $f_{post-ODCLK} = f_{VCO\ CLK} / 8$ <b>Note: Clock is gated if ODPLL is changed while the PLL is active.</b>



**Table 10-7. PLL Control 2 Register (PLLCTL2) Field Descriptions (continued)**

Bit	Field	Value	Description
8-0	SPR_AMOUNT		Spreading Amount $NV = (SPR\_AMOUNT + 1)/2048$ User and privileged mode (read): Privileged mode (write):
		0	$NV = 1/2048$
		1h	$NV = 2/2048$
		:	continues in sequence
		1FFh	$NV = 512/2048$

### 10.6.3 PLL Control 3 Register (PLLCTL3)

Figure 10-8 illustrates this register and Table 10-8 provides the bit descriptions.

**Figure 10-8. PLL Control 1 Register (PLLCTL3) [offset = FFFF E100h]**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ODPLL[2:0]			PLLDIV[4:0]				Reserved			REFCLKDIV[5:0]					
R/W-1			R/W-1Fh				R-0			R/W-2h					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLLMUL[15:0]															
R/W-5F00h															

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

**Table 10-8. PLL Control 3 Register (PLLCTL3) Field Descriptions**

Bit	Field	Value	Description
31-29	ODPLL	0h 1h : 7h	Internal PLL Output Divider $OD = ODPLL + 1$ $f_{post-ODCLK} = f_{VCO\ CLK} / OD$ User and privileged mode (read): Privileged mode (write): $f_{post-ODCLK} = f_{VCO\ CLK} / 1$ $f_{post-ODCLK} = f_{VCO\ CLK} / 2$ continues in sequence $f_{post-ODCLK} = f_{VCO\ CLK} / 8$ <b>Note: Clock is gated if ODPLL is changed while the PLL is active.</b>
28-24	PLLDIV	0 1h : 1Fh	PLL Output Clock Divider $R = PLLDIV + 1$ $f_{PLL\ CLK} = f_{post-ODCLK} / R$ User and privileged mode (read): Privileged mode (write): $f_{PLL\ CLK} = f_{post-ODCLK} / 1$ $f_{PLL\ CLK} = f_{post-ODCLK} / 2$ continues in sequence $f_{PLL\ CLK} = f_{post-ODCLK} / 32$
23-22	Reserved	0	Read returns 0. Writes have no effect.
21-16	REFCLK DIV	0 1h : 3Eh	Reference Clock Divider $NR = REFCLKDIV + 1$ (63 -- max) $f_{INT\ CLK} = f_{OSCIN} / NR$ User and privileged mode (read): Privileged mode (write): $f_{INT\ CLK} = f_{OSCIN} / 1$ $f_{INT\ CLK} = f_{OSCIN} / 2$ continues in sequence $f_{INT\ CLK} = f_{OSCIN} / 63$ <b>Note: Typically, this value should not be changed while the PLL is active.</b>

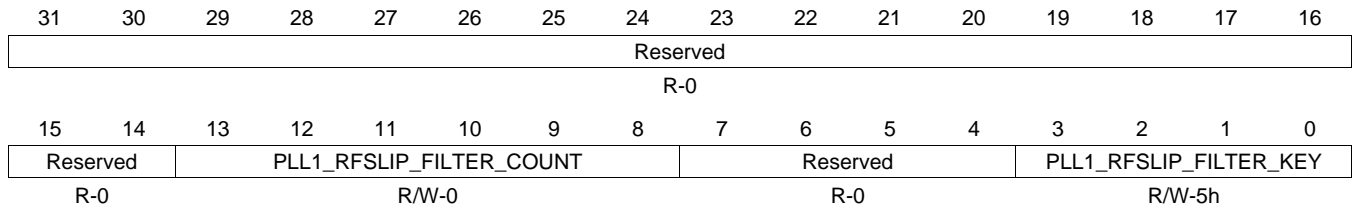
**Table 10-8. PLL Control 3 Register (PLLCTL3) Field Descriptions (continued)**

Bit	Field	Value	Description
15-0	PLLMUL		PLL Multiplication Factor Valid multiplication factors are from 1 to 256. For non-modulated mode: $NF = (PLLMUL / 256) + 1$ For modulated mode, MULMOD bit field is added as in <a href="#">Table 10-2</a> $f_{VCO\ CLK} = f_{INT\ CLK} \times NF$ User and privilege mode (read): Privilege mode (write):
		100h	$f_{VCO\ CLK} = f_{INT\ CLK} \times 1$
		180h	$f_{VCO\ CLK} = f_{INT\ CLK} \times 1.5$
		5C00h	$f_{VCO\ CLK} = f_{INT\ CLK} \times 93$
		:	continues in sequence
		FF00h	$f_{VCO\ CLK} = f_{INT\ CLK} \times 256$

**10.6.4 PLL1 Clock Slip Control Register (CLKSLIP)**

Figure 10-9 illustrates this register and Table 10-9 provides the bit descriptions.

**Figure 10-9. PLL Clock Slip Control Register (CLKSLIP) [offset = FFFF E170h]**



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

**Table 10-9. PLL Clock Slip Control Register (CLKSLIP) Field Descriptions**

Bit	Field	Value	Description
31-14	Reserved	0	Read returns 0. Writes have no effect.
13-8	PLL1_RFSLIP_FILTER_COUNT	0 1h 2h : 3Fh	Consecutive cycles of the HF LPO for which RFSLIP from PLL1 must be active in order to be recognized as a slip. User and privileged mode (read): Privileged mode (write): 0 Filtering is disabled. Every slip is recognized. 1h Filtering is enabled. Every slip is recognized. 2h Filtering is enabled. The slip must be at least 2 HF LPO cycles wide in order to be recognized as a slip. : continues in sequence 3Fh Filtering is enabled. The slip must be at least 63 HF LPO cycles wide in order to be recognized as a slip.
7-4	Reserved	0	Read returns 0. Writes have no effect.
3-0	PLL1_RFSLIP_FILTER_KEY	5h Fh other	Key to enable slip filtering for RFSLIP from PLL1. User and privileged mode (read): Privileged mode (write): 5h Filtering is disabled. Fh Test setting. Do not use. other Filtering is enabled.

### 10.6.5 General Purpose Register1 (GPREG1)

Figure 10-10 illustrates this register and Table 10-10 provides the bit descriptions.

**Figure 10-10. General Purpose Register1 (GPREG1) [offset = FFFF FFA0h]**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						PLL1_FBSLIP_FILTER_COUNT						PLL1_FBSLIP_FILTER_KEY			
R-0						R/W-0						R/W-5h			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
other fields (see GPREG1; see Architecture Users Guide)															

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

**Table 10-10. General Purpose Register1 (GPREG1) Field Descriptions**

Bit	Field	Value	Description
31-26	Reserved	0	Read returns 0. Writes have no effect.
25-20	PLL1_FBSLIP_FILTER_COUNT	0 1h 2h : 3Fh	Consecutive cycles of the HF LPO for which FBSLIP from PLL1 must be active in order to be recognized as a slip . User and privileged mode (read): Privileged mode (write): 0 Filtering is disabled. Every slip is recognized. 1h Filtering is enabled. Every slip is recognized. 2h Filtering is enabled. The slip must be at least 2 HF LPO cycles wide in order to be recognized as a slip. : continues in sequence 3Fh Filtering is enabled. The slip must be at least 63 HF LPO cycles wide in order to be recognized as a slip.
19-16	PLL1_FBSLIP_FILTER_KEY	5h Fh other	Key to enable slip filtering for FBSLIP from PLL1 User and privileged mode (read): Privileged mode (write): 5h Filtering is disabled. Fh Test setting. Do not use. other Filtering is enabled.
15-0	other bit fields		see GPREG1 for other bit fields not related to the PLL

### 10.6.6 PLL Modulation Depth Measurement Control Register (SSWPLL1)

Figure 10-11 illustrates this register and Table 10-11 provides the bit descriptions.

**Figure 10-11. SSW PLL BIST Control Register 1 (SSWPLL1) [offset = 24h]**

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								
CAPTURE_WINDOW_INDEX															
R/W-0															
7	6	5	4	3	2	1	0								
Reserved		COUNTER_READ_READY	COUNTER_RESET	COUNTER_EN	TAP_COUNTER_DIS			EXT_COUNTER_EN							
R-0		R-0	R/W-1	R/W-0	R/W-0										

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

**Table 10-11. SSW PLL BIST Control Register 1 (SSWPLL1) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-8	CAPTURE_WINDOW_INDEX	0-FFh	The capture counter present in the PLL wrapper will count the PLL clock edges when the current modulation phase capture window value is equal to these bits. Should be set equal to NR.
7	Reserved	0	Read returns 0. Writes have no effect.
6	COUNTER_READ_READY	0 1	Counter read ready. Indicates that SSW_CAPTURE_COUNT (SSWPLL2) and SSW_CLKOUT_COUNT (SSWPLL3) can be read. Counter registers in SSWPLL2 and SSWPLL3 are not ready to read. Counter registers in SSWPLL2 and SSWPLL3 are ready to read.
5	COUNTER_RESET	0 1	Counter reset. If EXT_COUNTER_EN = 0, COUNTER_RESET resets SSW_CAPTURE_COUNT (SSWPLL2) and SSW_CLKOUT_COUNT (SSWPLL3). If EXT_COUNTER_EN = 1, this bit is ignored. No impact to counters If the EXT_COUNTER_EN bit is 0, then counters SSW_CAPTURE_COUNT and SSW_CLKOUT_COUNT will be held in the reset state. If EXT_COUNTER_EN bit is 1, then this bit will be ignored by the PLL wrapper.
4	COUNTER_EN	0 1	Counter enable. If EXT_COUNTER_EN = 0, COUNTER_EN initializes the modulation depth measurement. (In this mode, the disable is set to occur automatically.) If EXT_COUNTER_EN = 1, the counters are enabled/disabled with COUNTER_EN. If EXT_COUNTER_EN = 0, COUNTER_EN = 0 indicates that the counters are inactive. If EXT_COUNTER_EN = 1, COUNTER_EN = 0 disables the counters. If EXT_COUNTER_EN = 0, COUNTER_EN = 0 indicates that the counters are still active. If EXT_COUNTER_EN = 1, COUNTER_EN = 0 enables the counters.

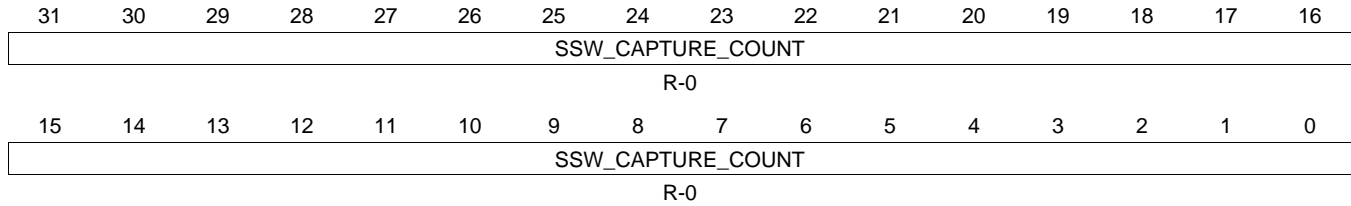
**Table 10-11. SSW PLL BIST Control Register 1 (SSWPLL1) Field Descriptions (continued)**

Bit	Field	Value	Description
3-1	TAP_COUNTER_DIS		The value in this register is used to program a particular bit in CLKOUT counter. When that particular bit in CLKOUT counter becomes 1, then both the CLKOUT counter and the CAPTURE counter will stop counting when EXT_COUNTER_EN = 0. When EXT_COUNTER_EN = 1, this bit field is not used.
		0	Bit 16 of CLKOUT counter is selected. When this bit is set and the modulation period finishes, the counters are disabled and READ_READY_FLAG is set
		1h	Bit 18 of CLKOUT counter is selected.
		2h	Bit 20 of CLKOUT counter is selected.
		3h	Bit 22 of CLKOUT counter is selected.
		4h	Bit 24 of CLKOUT counter is selected.
		5h	Bit 26 of CLKOUT counter is selected.
		6h	Bit 28 of CLKOUT counter is selected.
0	EXT_COUNTER_EN	0	Modulation Depth Measurement mode
		1	Frequency Measurement mode

### 10.6.7 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 10-12](#) and described in [Table 10-12](#).

**Figure 10-12. SSW PLL BIST Control Register 2 (SSWPLL2) [offset = 28h]**



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

**Table 10-12. SSW PLL BIST Control Register 2 (SSWPLL2) Field Descriptions**

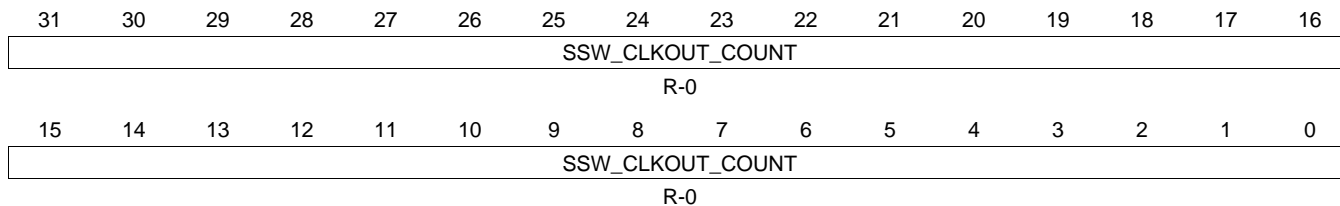
Bit	Field	Value	Description
31-0	SSW_CAPTURE_COUNT	0-FFFF FFFFh	Capture count. This register returns the value of the capture count. When EXT_COUNTER_EN = 0, this counter increments within a fixed modulation window. When EXT_COUNTER_EN = 1, this counter increments based upon the oscillator.



### 10.6.8 SSW PLL BIST Control Register 3 (SSWPLL3)

This is observation register used to log counter value for CLKOUT counter inside PLL wrapper. The SSWPLL3 register is shown in [Figure 10-13](#) and described in [Table 10-13](#).

**Figure 10-13. SSW PLL BIST Control Register 3 (SSWPLL3) [offset = 2Ch]**



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

**Table 10-13. SSW PLL BIST Control Register 3 (SSWPLL3) Field Descriptions**

Bit	Field	Value	Description
31-0	SSW_CAPTURE_COUNT	0-FFFF FFFFh	Value of CLKout count register. This counter increments based upon the PLL output (prior to the R-divider).

### 10.6.9 LPO/Clock Monitor Control Register (LPOMONCTL)

This register is shown in [Figure 10-14](#) and described in [Table 10-14](#).

**Figure 10-14. LPO/Clock Monitor Control Register (LPOMONCTL) [offset = 88h]**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						BIAS ENABLE		Reserved						OSCFRQCONF IGCNT	
R-0						R/W-1		R-0						R/W-0	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			HFTRIM				Reserved			LFTRIM					
R-0			R/W-10h				R-0			R/W-10h					

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

**Table 10-14. LPO/Clock Monitor Control Register (LPOMONCTL) Field Descriptions**

Bit	Field	Value	Description
31-25	Reserved	0	Read returns 0. Writes have no effect.
24	BIAS ENABLE	0	Bias enable. Reset by nPORRST. The bias circuit inside the low-power oscillator (LPO) is disabled.
		1	The bias circuit inside LPO is enabled.
23-17	Reserved	0	Read returns 0. Writes have no effect.
16	OSCFRQCO NFIGCNT	0	Configures the counter based on OSC frequency. OSC freq is <= 20MHz
		1	OSC freq is > 20MHz and <= 80MHz
15-13	Reserved	0	Read returns 0. Writes have no effect.

**Table 10-14. LPO/Clock Monitor Control Register (LPOMONCTL) Field Descriptions (continued)**

Bit	Field	Value	Description																																																																
12-8	HFTRIM		<p>High frequency oscillator trim value. This four-bit value is used to center the HF oscillator's frequency. Reset by nPORRST.</p> <p><b>Caution: This value should only be changed when the HF oscillator is not the source for a clock domain, otherwise a system failure could result.</b></p> <p><b>HF TRIM values below reflect for the F021 process.</b></p> <p>The values below are the ratio, <math>f / f_0</math>, expressed as a percent:</p> <table> <tr><td>0</td><td>29.52</td></tr> <tr><td>1h</td><td>34.24</td></tr> <tr><td>2h</td><td>38.85</td></tr> <tr><td>3h</td><td>43.45</td></tr> <tr><td>4h</td><td>47.99</td></tr> <tr><td>5h</td><td>52.55</td></tr> <tr><td>6h</td><td>57.02</td></tr> <tr><td>7h</td><td>61.46</td></tr> <tr><td>8h</td><td>65.92</td></tr> <tr><td>9h</td><td>70.17</td></tr> <tr><td>Ah</td><td>74.55</td></tr> <tr><td>Bh</td><td>78.92</td></tr> <tr><td>Ch</td><td>83.17</td></tr> <tr><td>Dh</td><td>87.43</td></tr> <tr><td>Eh</td><td>91.75</td></tr> <tr><td>Fh</td><td>95.89</td></tr> <tr><td>10h</td><td>100.00% Default at Reset.</td></tr> <tr><td>11h</td><td>104.09</td></tr> <tr><td>12h</td><td>108.17</td></tr> <tr><td>13h</td><td>112.32</td></tr> <tr><td>14h</td><td>116.41</td></tr> <tr><td>15h</td><td>120.67</td></tr> <tr><td>16h</td><td>124.42</td></tr> <tr><td>17h</td><td>128.38</td></tr> <tr><td>18h</td><td>132.24</td></tr> <tr><td>19h</td><td>136.15</td></tr> <tr><td>1Ah</td><td>140.15</td></tr> <tr><td>1Bh</td><td>143.94</td></tr> <tr><td>1Ch</td><td>148.02</td></tr> <tr><td>1Dh</td><td>151.80</td></tr> <tr><td>1Eh</td><td>155.50</td></tr> <tr><td>1Fh</td><td>159.35</td></tr> </table>	0	29.52	1h	34.24	2h	38.85	3h	43.45	4h	47.99	5h	52.55	6h	57.02	7h	61.46	8h	65.92	9h	70.17	Ah	74.55	Bh	78.92	Ch	83.17	Dh	87.43	Eh	91.75	Fh	95.89	10h	100.00% Default at Reset.	11h	104.09	12h	108.17	13h	112.32	14h	116.41	15h	120.67	16h	124.42	17h	128.38	18h	132.24	19h	136.15	1Ah	140.15	1Bh	143.94	1Ch	148.02	1Dh	151.80	1Eh	155.50	1Fh	159.35
0	29.52																																																																		
1h	34.24																																																																		
2h	38.85																																																																		
3h	43.45																																																																		
4h	47.99																																																																		
5h	52.55																																																																		
6h	57.02																																																																		
7h	61.46																																																																		
8h	65.92																																																																		
9h	70.17																																																																		
Ah	74.55																																																																		
Bh	78.92																																																																		
Ch	83.17																																																																		
Dh	87.43																																																																		
Eh	91.75																																																																		
Fh	95.89																																																																		
10h	100.00% Default at Reset.																																																																		
11h	104.09																																																																		
12h	108.17																																																																		
13h	112.32																																																																		
14h	116.41																																																																		
15h	120.67																																																																		
16h	124.42																																																																		
17h	128.38																																																																		
18h	132.24																																																																		
19h	136.15																																																																		
1Ah	140.15																																																																		
1Bh	143.94																																																																		
1Ch	148.02																																																																		
1Dh	151.80																																																																		
1Eh	155.50																																																																		
1Fh	159.35																																																																		
7-5	Reserved	0	Read returns 0. Writes have no effect.																																																																

**Table 10-14. LPO/Clock Monitor Control Register (LPOMONCTL) Field Descriptions (continued)**

Bit	Field	Value	Description
4-0	LFTRIM		<p>Low frequency oscillator trim value. This four-bit value is used to center the LF oscillator's frequency. Reset by nPORRST.</p> <p><b>Caution: This value should only be changed when the LF oscillator is not the source for a clock domain, otherwise a system failure could result.</b></p> <p><b>LF TRIM values below reflect for the F021 process.</b></p> <p>The values below are the ratio, <math>f / f_0</math>, expressed as a percent:</p>
		0	20.67
		1h	25.76
		2h	30.84
		3h	35.90
		4h	40.93
		5h	45.95
		6h	50.97
		7h	55.91
		8h	60.86
		9h	65.78
		Ah	70.75
		Bh	75.63
		Ch	80.61
		Dh	85.39
		Eh	90.23
		Fh	95.11
		10h	100.00% Default at Reset
		11h	104.84
		12h	109.51
		13h	114.31
		14h	119.01
		15h	123.75
		16h	128.62
		17h	133.31
		18h	138.03
		19h	142.75
		1Ah	147.32
		1Bh	152.02
		1Ch	156.63
		1Dh	161.38
		1Eh	165.90
		1Fh	170.42

### 10.6.10 Clock Test Register (CLKTEST)

This register is shown in [Figure 10-15](#) and described in [Table 10-15](#).

**Figure 10-15. Clock Test Register (CLKTEST) [offset = 8Ch]**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
Reserved					TEST	RANGE DET CTRL	RANGE DET ENASS EL	Reserved					CLK_TEST_EN				
R-0					R/W-0	R/W-0	R/W-0	R-0					R/W-Ah				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Reserved					SEL_GIO_PIN			Reserved					SEL_ECP_PIN				
R-0					R/W-0			R-0					R/W-0				

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

**Table 10-15. Clock Test Register (CLKTEST) Field Descriptions**

Bit	Field	Value	Description
31-27	Reserved	0	Read returns 0. Writes have no effect.
26	TEST		This bit is used for test purposes. It must be written to 0.
25	RANGEDET CTRL	0	The clock monitor range detection circuitry (RANGEDETECTENABLE) is disabled.
		1	The clock monitor range detection circuitry (RANGEDETECTENABLE) is enabled.
24	RANGEDET ENASSSEL	0	Selects range detection enable. This bit resets synchronously on SYS_nRST. 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]).
23-20	Reserved	0	Read returns 0. Writes have no effect.
19-16	CLK_TEST_ EN		Clock test enable. This bit enables the clock going to the ECLK pin. 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. These bits needs to asynchronously reset.
		5h	Clock going to ECLK pin is enabled.
		Others	Clock going to ECLK pin is disabled.
15-12	Reserved	0	Read returns 0. Writes have no effect.
11-8	SEL_GIO_PI N		Select the clock source valid signal or clock source at functional GIO pin This bit field is device dependent please consult the device-specific data sheet
7-5	Reserved	0	Read returns 0. Writes have no effect.
4-0	SEL_ECP_PI N		Select the clock at ECP Pin. This bit field is device dependent please consult the device-specific data sheet.

**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.

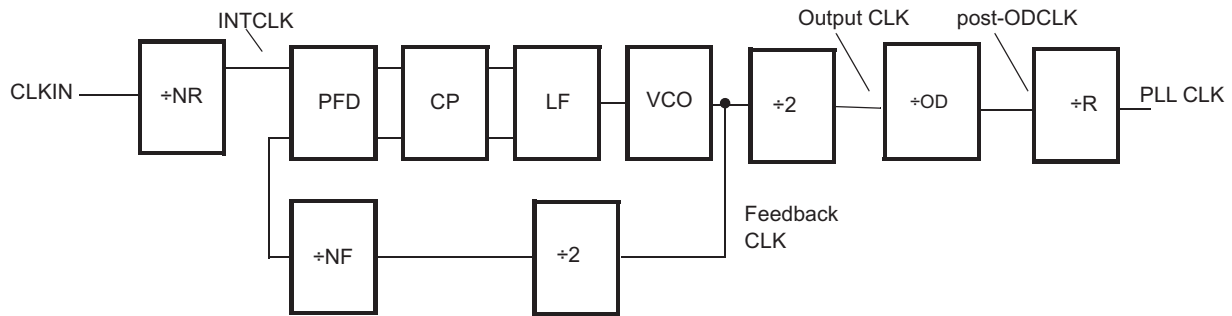
## 10.7 Phase-Locked Loop Theory of Operation

The PLL block consists of six logical sub-blocks:

- Phase-Frequency Detector (PFD)
- Charge Pump (CP)
- Loop Filter (LF)
- Voltage-Controlled Oscillator (VCO)
- Frequency Modulation
- Slip Detector

Figure 10-16 illustrates the sub-blocks in a basic PLL circuit. The VCO adjusts its frequency until the two signals into the PFD have the same phase and frequency. The feedback path (from VCO to PFD) divides the frequency of the feedback signal by  $2 \cdot NF$ ; this feedback divider requires the VCO to generate a frequency  $2 \cdot NF$  times greater than the internal frequency ( $OSCIN/NR$ ). In the forward path (from VCO to PLL CLK), the  $/2$  block creates a clean duty cycle.

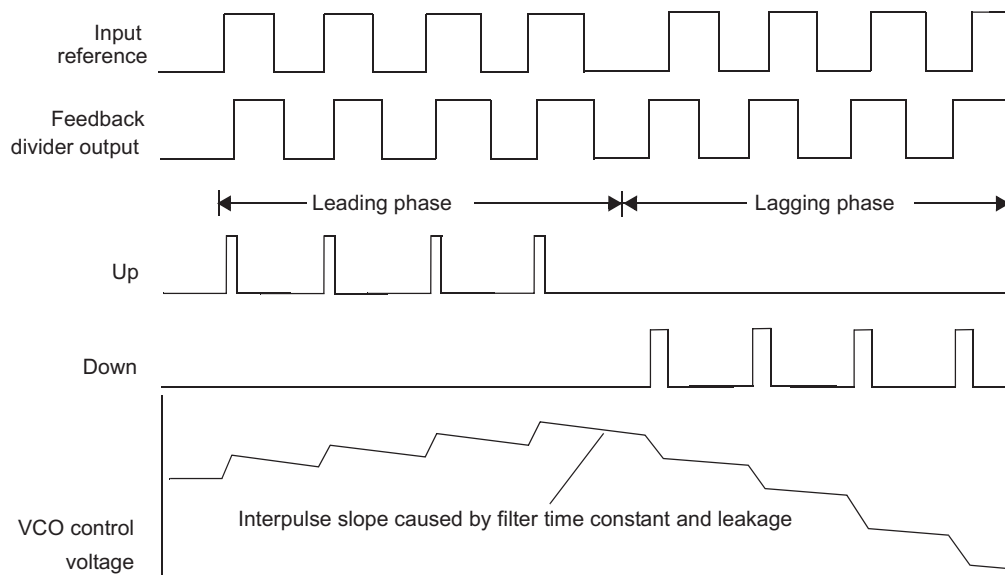
Figure 10-16. Basic PLL Circuit



### 10.7.1 Phase-Frequency Detector

The phase-frequency detector (PFD) compares the input reference phase/frequency to the phase/frequency of the feedback divider and generates two signals: an *up* pulse and a *down* pulse that drive a charge pump. The resulting charge, when integrated by the circuit at the LF pin, provides a VCO control voltage, as shown in Figure 10-17.

Figure 10-17. PFD Timing



The width of the up pulse and the down pulse depends on the difference in phase between the two inputs. For example, when the reference input leads the feedback input by 10 ns, then an up pulse of approximately 10 ns is generated (see [Figure 10-17](#)). 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  $2 \cdot NF$  times the reference frequency.

The reference feedback frequency is based upon the VCO frequency and the feedback divider. Fractional multiplication is achieved by changing the feedback divider real-time in order to create the fractional multiplication. As an example, if a multiplier of 100.5 is selected, the feedback divider divides by 100 and 101 in equal proportions; in this case, the PLLMUL bit field would be programmed as 99.5 (0x6380). This fractional multiplication is useful when trying to achieve final frequencies that are non-integer to the input frequency (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.

### 10.7.2 Charge Pump and Loop Filter

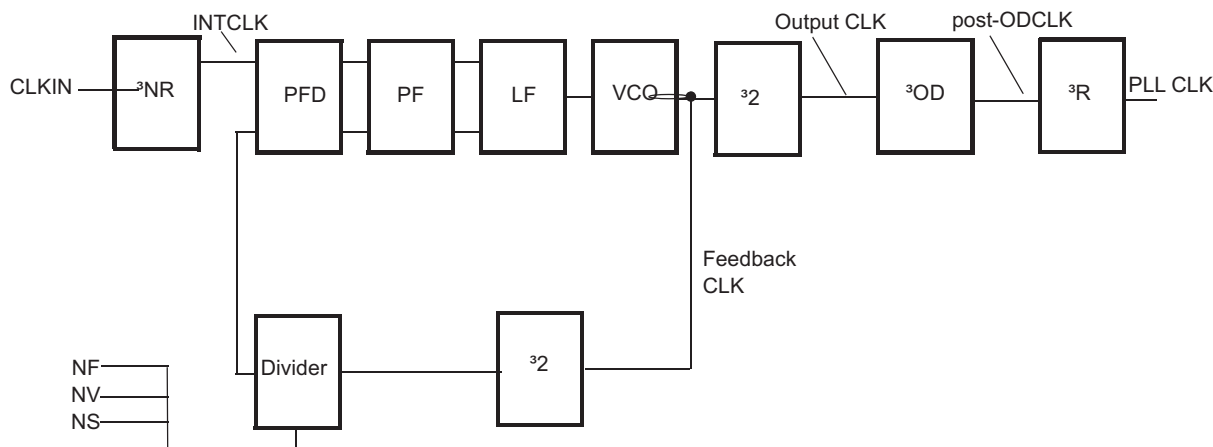
The charge pump (CP) add or remove charge from the loop filter based on the pulses coming from the phase-frequency detector (PFD).

Two components of the filter output signal are summed together: an integral component and a proportional component. The integral component maintains a DC level going to the VCO to set its frequency, and the proportional component makes the VCO track changes in phase to minimize jitter. The capacitors and resistors required for the filter are integrated in silicon.

### 10.7.3 Voltage-Controlled Oscillator

The output frequency of the VCO is proportional to its input control voltage, which is generated by the charge pump via the integrated loop filter. If the VCO oscillates too slowly, the feedback phase begins to lag the reference phase at the PFD, which increases the control voltage at the VCO. Conversely, if the VCO oscillates too fast, the feedback phase begins to lead the reference phase at the PFD, which decreases the control voltage at the VCO. These two actions keep the VCO running at the correct frequency multiple of the reference.

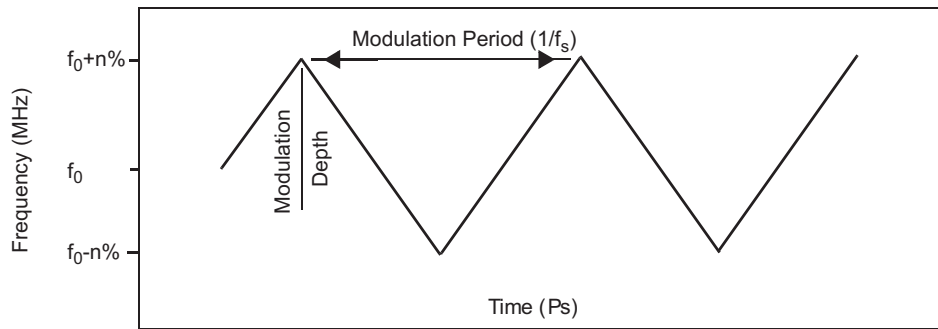
**Figure 10-18. PLL Modulation Block Diagram**



### 10.7.4 Frequency Modulation

The output clock of the PLL changes frequency in a controlled way, centered around the unmodulated output frequency. The modulation block directly modulates the VCO frequency at the loop filter, and creates the triangular frequency modulation (see [Figure 10-19](#)).

**Figure 10-19. Frequency vs. Time**





## 10.8 Programming Example

This section provides an example of how to program the PLL. For non-modulation settings, the PLLCTL1 and PLLCTL2 settings from 130nm process devices can be used without modification.

Suppose that using a 20MHz crystal, the application requires a

- 180MHz GCLK (and HCLK) frequency
- 100 kHz spreading frequency
- 0.5% spreading depth.

1. Choose an NR and NS such that

$$\frac{f_{CLKIN}}{NR \times f_s} \geq 40 \quad (14)$$

$$f_s \equiv \frac{f_{CLKIN}}{2 \times NR \times NS} \quad (15)$$

$$2 \times NS = \frac{f_{CLKIN}}{NR \times f_s} \geq 40 \quad (16)$$

- (NR,NS)={(5,20), (4,25), (2,50), (1,100)}
  - Either NR = 5 and NS = 20 or NR = 4 and NS = 25 are reasonable. Another choice, (NR, NS) = (3, 33) is possible if the modulation frequency can vary from 100KHz.
2. Choose Output CLK frequency as integer divider of output frequency near to 330MHz. Output CLK frequency should not exceed 550MHz or fall below 110MHz. The integer values for 180MHz are 360MHz, or 540MHz. 360MHz is close to the target frequency of 330MHz and we use this frequency.

3. In this case, either of the below equations are suitable choices for getting to 360MHz. Choose NR = 5, NS = 20 and set NF = 72.

$$\frac{f_{CLKIN}}{NR} = \frac{20[MHz]}{4} = 5[MHz] \text{ or } \frac{f_{CLKIN}}{NR} = \frac{20[MHz]}{5} = 4[MHz] \quad (17)$$

4. Select the output divider OD so that the post-ODCLK frequency does not exceed the maximum frequency of output divider R (device specific frequency). In this case, choose OD = 2 and R = 1.

5. Compute the divider value NV so that NV = 0.036.

$$Depth = \frac{0.5}{100} = \frac{NV}{NF} \times \frac{NS}{2} = \frac{NV}{72} \times \frac{20}{2} \quad (18)$$

6. If it is important to maintain the same average frequency in modulation as in non-modulation, either NF should be modified OR program the MULMOD bit field. The modulation fields create a multiplier offset equal to:

$$\Delta NF = \frac{NV \times NS}{2} \quad (19)$$

If using MULMOD[8:0], then:

$$\Delta NF = \frac{MULMOD[8...0]}{256} = \frac{NV \times NS}{2} = \frac{0.036 \times 20}{2} \quad (20)$$

$$MULMOD[8...0] = \frac{0.036 \times 20}{2} \times 256 = 92.16 \quad (21)$$

MULMOD will be set to 92.

7. Convert the PLL parameters into bit field values:

- NR = 5 implies that REFCLKDIV[5:0] = 4
  - NS = 20 implies that SPRATE[8:0] = 19 = 0x13
  - NF = 72 implies that PLLMUL[15:0] = 0x4700
  - OD = 2 implies that ODPLL[2:0] = 1
  - R = 1 implies that PLLDIV[4:0] = 0
  - NV = 0.036 implies that SPRAMOUNT = 73 = 0x49
  - MULMOD = 92 = 0x5C
8. Setting only these fields (that is, not BPOS, ROF, or ROS) yields  
PLLCTL1 = 0x00044700  
PLLCTL2 = 0x04C5C249  
When FM ENA is turned on, PLLCTL2 = 0x84C5C249  
The Output CLK is centered in the range from 110MHz to 550MHz at 360MHz.  
NF = 72 falls within the multiplier range from 1 - 256  
OD is selected so that post-ODCLK meets the device specification.



---

---

## ***Dual-Clock Comparator (DCC) Module***

---

---

This chapter describes the dual-clock comparator (DCC) module.

<b>Topic</b>	<b>Page</b>
<b>11.1 Introduction .....</b>	<b>410</b>
<b>11.2 Module Operation .....</b>	<b>411</b>
<b>11.3 Clock Source Selection for Counter0 and Counter1 .....</b>	<b>414</b>
<b>11.4 DCC Control Registers .....</b>	<b>415</b>

## 11.1 Introduction

The microcontroller includes two dual-clock comparator (DCC1, DCC2) modules. The primary purpose of a DCC module is to measure the frequency of a clock signal using a second clock signal as a reference. An application can use this capability to further enhance the system safety.

### 11.1.1 Main Features

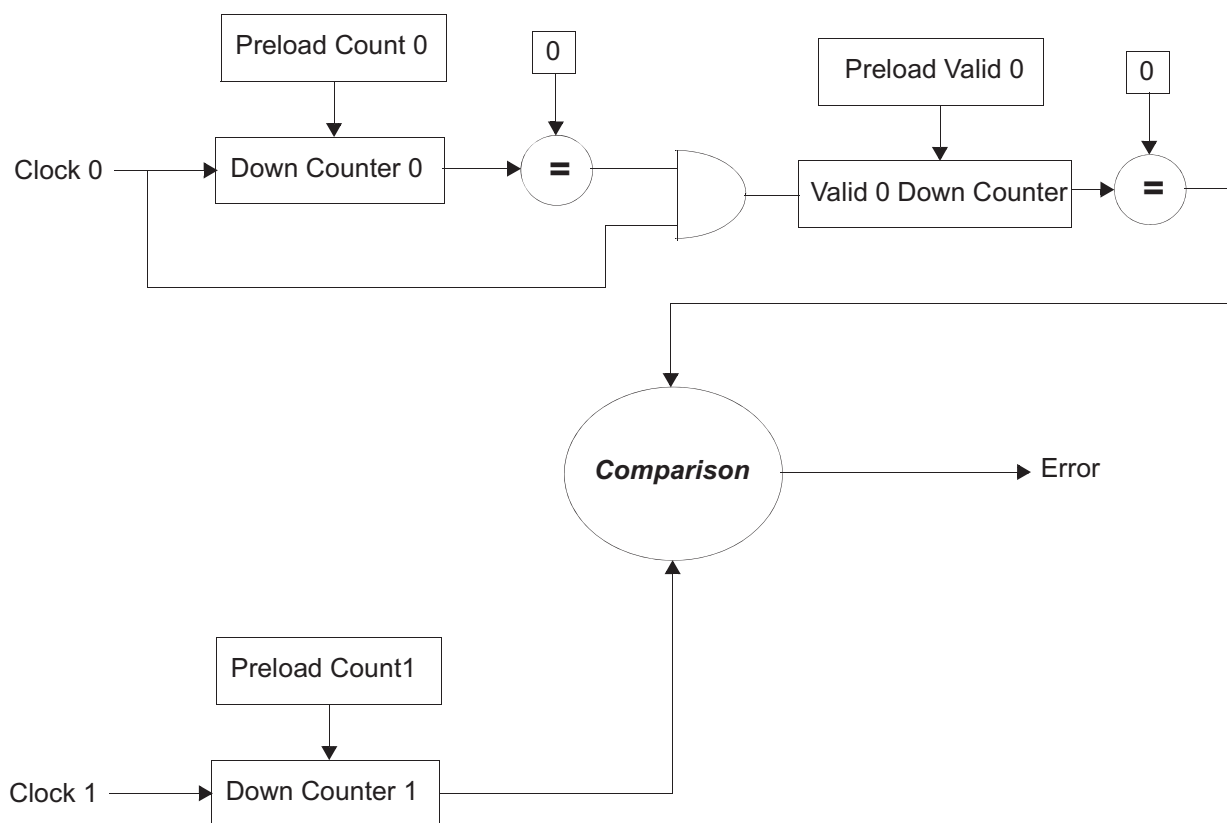
The main features of each of the DCC modules are:

- Allows application to ensure that a fixed ratio is maintained between frequencies of two clock signals
- Supports the definition of a programmable tolerance window in terms of number of reference clock cycles
- Supports continuous monitoring without requiring application intervention
- Alternatively can be used in a single-sequence mode for spot measurements
- Flexible clock source selection for counter0 and counter1 resulting in several specific use cases

### 11.1.2 Block Diagram

Figure 11-1 illustrates the main concept of the DCC module.

**Figure 11-1. DCC Block Diagram**



## 11.2 Module Operation

As shown in [Figure 11-1](#) the DCC contains two counters – counter0 and counter1, which are driven by two signals – clock0 and clock1. The application programs the seed values for both these counters. The application also configures the tolerance window time by configuring the valid counter for clock0.

The counter0 and counter1 both start counting simultaneously when the DCC is enabled. When counter0 counts down to zero, it triggers the count down of the tolerance window counter called valid0.

The seed values of counter0 and counter1 are selected such that if the actual frequencies of clock0 and clock1 are equal to their expected frequencies, then the counter1 will reach zero either at the same time as counter0 or during the count down of the tolerance window counter.

### 11.2.1 Error Conditions

An error condition is generated by any one of the following:

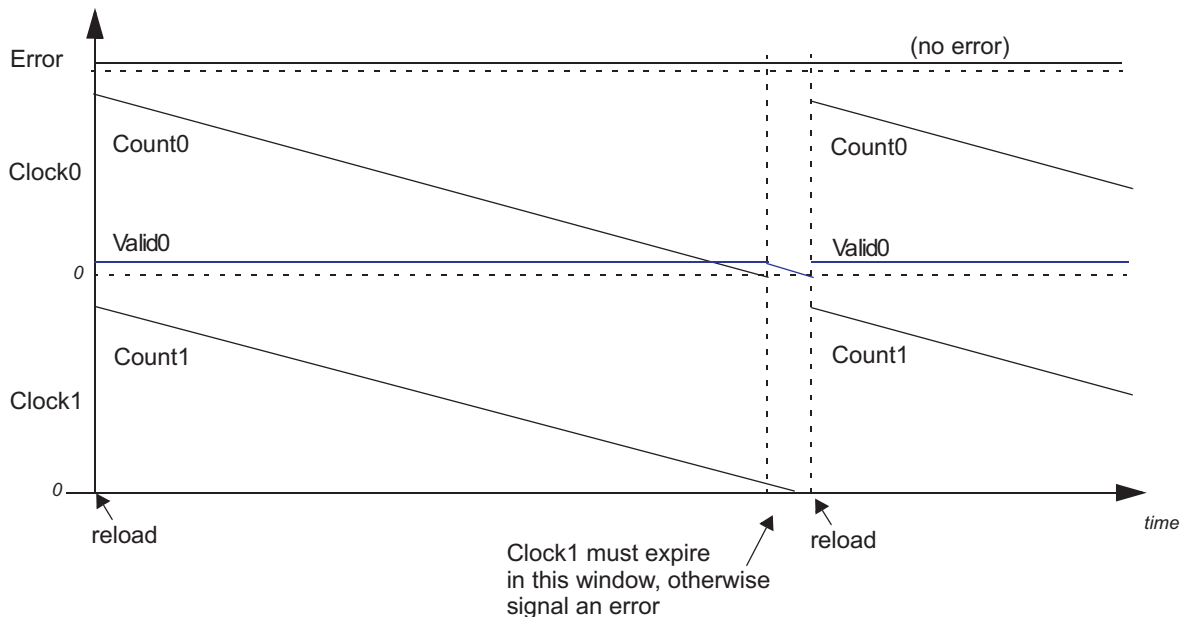
1. Counter1 counts down to 0 before Counter0 reaches 0
2. Counter1 does not reach 0 even when Counter0 and Valid0 have both reached 0
3. Clock0 is not present
4. Clock1 is not present

Any error causes the counters to stop counting. An application may then read out the counter values to help determine what caused the error.

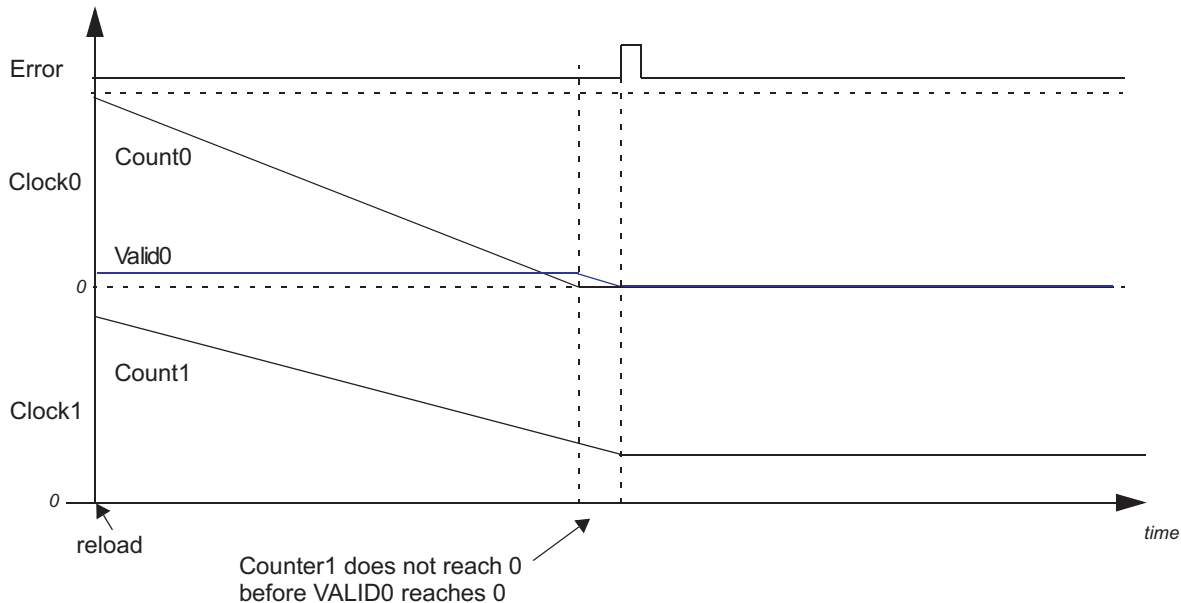
The counters can also get reloaded with the seed values and continue counting down under the following conditions:

- The module is reset or restarted by the application, OR
- Counter0, Valid 0 and Counter1 all reach 0 without any error

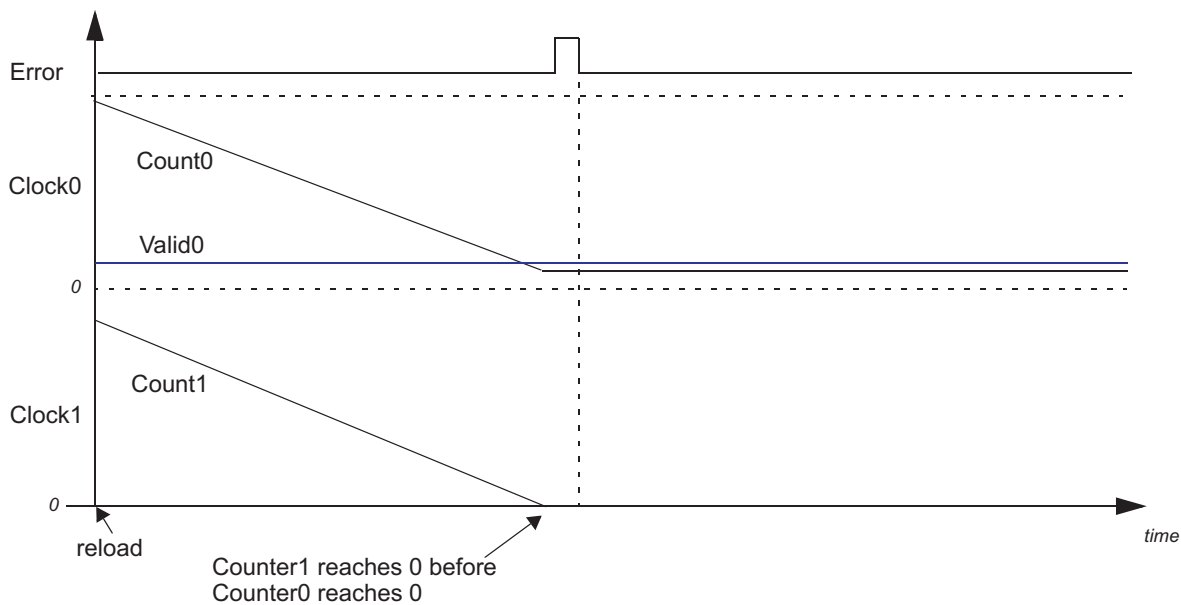
**Figure 11-2. Counter Relationship**



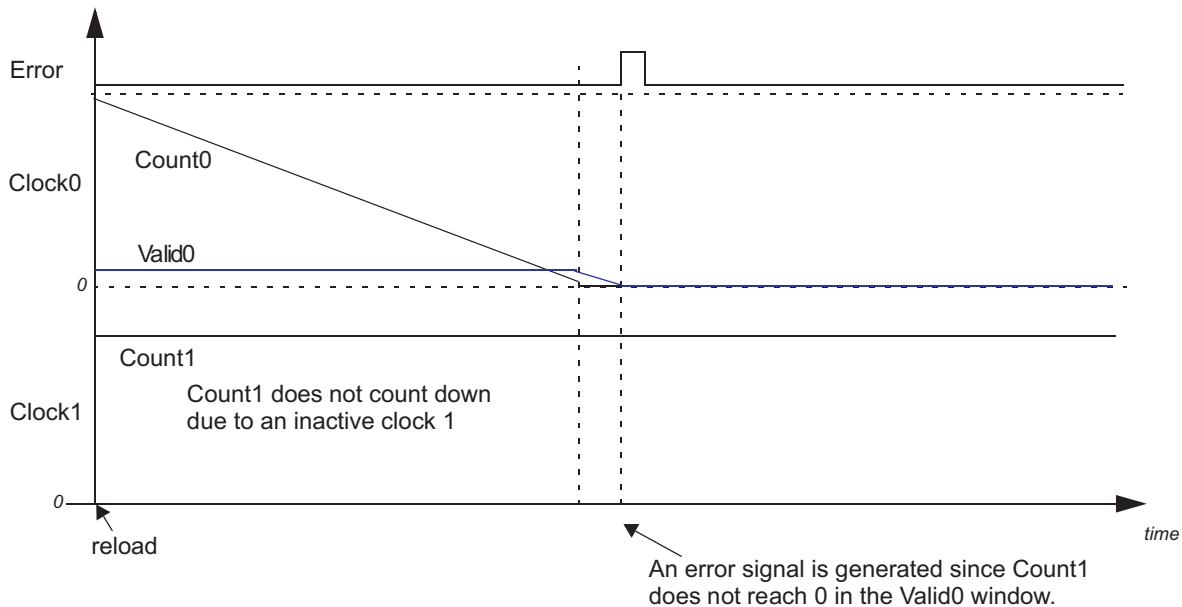
**Figure 11-3. Clock1 slower than Clock0 results in an error and stops counting**



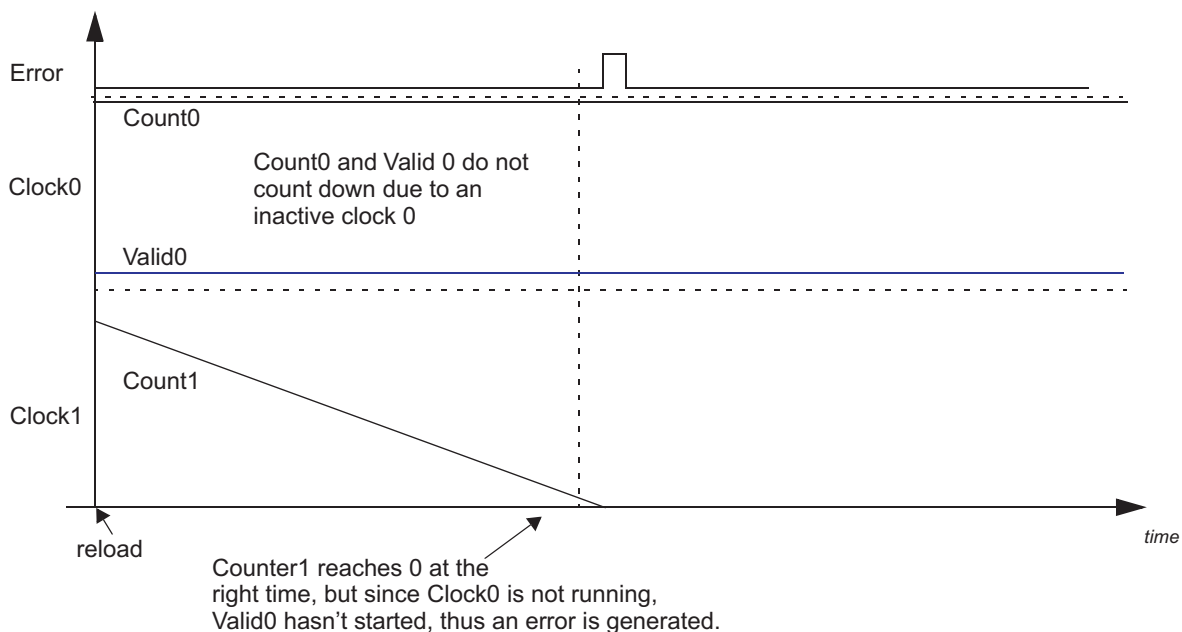
**Figure 11-4. Clock1 faster than Clock0 results in an error and stops counting**



**Figure 11-5. Clock1 not present results in an error and stops counting**



**Figure 11-6. Clock0 not present results in an error and stops counting**



### 11.2.2 Single-Shot Operation Mode

The DCC module can be programmed to count down one time by enabling the single-shot mode. In this mode, the DCC stops operating when the down counter0 and the valid counter0 reach 0.

At the end of one sequence of counting down in this single-shot mode, the DCC gets disabled automatically, which prevents further counting.

If there is no error generated at the end of the sequence, then the DONE status flag is set and a DONE interrupt is generated. The application must clear the DONE flag before restarting the DCC.

If an error is generated at the end of the sequence, then the ERROR status flag is set. The application must clear the ERROR status flag before restarting the DCC.



### 11.3 Clock Source Selection for Counter0 and Counter1

There are two DCC modules on the microcontrollers. Each DCC module has two counters. The table below shows the options for selecting the clock sources for both counters on each of the DCC modules.

#### 11.3.1 Clock Sources for DCC1

**Table 11-1. Clock Source Selection for DCC1 Counter0**

CNT0 CLKSRC[3-0]	Clock Name	Clock Description
0101	HF LPO	High-Frequency Output of Internal Low-Power Oscillator
1010	TCK	Test Clock
others	OSCIN	Main oscillator input

**Table 11-2. Clock Source Selection for DCC1 Counter1**

KEY[3-0]	CNT1 CLKSRC[3-0]	Clock Name	Clock Description
0xA	0x0	PLL1 Output	Output from PLL#1
	0x1	PLL2 Output	Output from PLL#2
	0x2	LF LPO	Low-Frequency output of internal Low-Power Oscillator
	0x3	HF LPO	High-Frequency output of internal Low-Power Oscillator
	0x4	Reserved	Reserved for TI use only
	0x5	EXTCLKIN1	External clock input source # 1
	0x6	EXTCLKIN2	External clock input source # 2
	0x7	Reserved	Reserved for TI use only
	0x8 – 0xF	VCLK	Peripheral bus clock
others	don't care	N2HET1[31]	Signal output on the N2HET1[31] signal by the N2HET1 module, either by the N2HET1 program execution, or by the application toggling the N2HET1[31] pin as general-purpose output.

CNT0 CLKSRC is a control field in the CNT0 CLKSRC register. KEY and CNT1 CLKSRC are control fields in the CNT1 CLKSRC register.

#### 11.3.2 Clock Sources for DCC2

**Table 11-3. Clock Source Selection for DCC2 Counter0**

CNT0 CLKSRC[3-0]	Clock Name	Clock Description
1010	TCK	Test Clock
others	OSCIN	Main oscillator input

**Table 11-4. Clock Source Selection for DCC2 Counter1**

KEY[3-0]	CNT1 CLKSRC[3-0]	Clock Name	Clock Description
0xA	0x0 – 0x7	Reserved	Invalid configuration
	0x8 – 0xF	VCLK	Peripheral bus clock
others	don't care	N2HET2[0]	Signal output on the N2HET2[0] signal by the N2HET1 module, either by the N2HET2 program execution, or by the application toggling the N2HET2[0] pin as general-purpose output.

CNT0 CLKSRC is a control field in the CNT0 CLKSRC register. KEY and CNT1 CLKSRC are control fields in the CNT1 CLKSRC register.

## 11.4 DCC Control Registers

This section describes the dual-clock comparator (DCC) module control and status registers. The registers support 8-bit, 16-bit or 32-bit writes and are aligned on a word (32-bit) boundary. The table shows address offsets from the module base address.

The microcontrollers have two DCC modules. The base address for DCC1 registers is FFFF EC00h and the base address for DCC2 registers is FFFF F400h.

**Table 11-5. DCC Registers**

Offset	Acronym	Register Description	Section
00	DCCCTRL	DCC Global Control Register	<a href="#">Section 11.4.1</a>
04h	DCCREV	DCC Revision Id Register	<a href="#">Section 11.4.2</a>
08h	DCCNT0SEED	DCC Counter0 Seed Register	<a href="#">Section 11.4.3</a>
0Ch	DCCVALID0SEED	DCC Valid0 Seed Register	<a href="#">Section 11.4.4</a>
10h	DCCNT1SEED	DCC Counter1 Seed Register	<a href="#">Section 11.4.5</a>
14h	DCCSTAT	DCC Status Register	<a href="#">Section 11.4.6</a>
18h	DCCDCNT0	DCC Counter0 Value Register	<a href="#">Section 11.4.7</a>
1Ch	DCCVALID0	DCC Valid0 Value Register	<a href="#">Section 11.4.8</a>
20h	DCCDCNT1	DCC Counter1 Value Register	<a href="#">Section 11.4.9</a>
24h	DCCNT1CLKSRC	DCC Counter1 Clock Source Selection Register	<a href="#">Section 11.4.10</a>
28h	DCCNT0CLKSRC	DCC Counter0 Clock Source Selection Register	<a href="#">Section 11.4.11</a>

### 11.4.1 DCC Global Control Register (DCCGCTRL)

Figure 11-7 and Table 11-6 describe the DCC Global Control register.

**Figure 11-7. DCC Control Register (DCCGCTRL) [offset = 00]**

31	Reserved												16		
R-0															
15	12	11	8	7	4	3							0		
DONE INT ENA				SINGLE SHOT				ERR ENA				DCC ENA			
R/W-5h				R/W-5h				R/W-5h				R/W-5h			

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

**Table 11-6. DCC Control Register (DCCGCTRL) Field Descriptions**

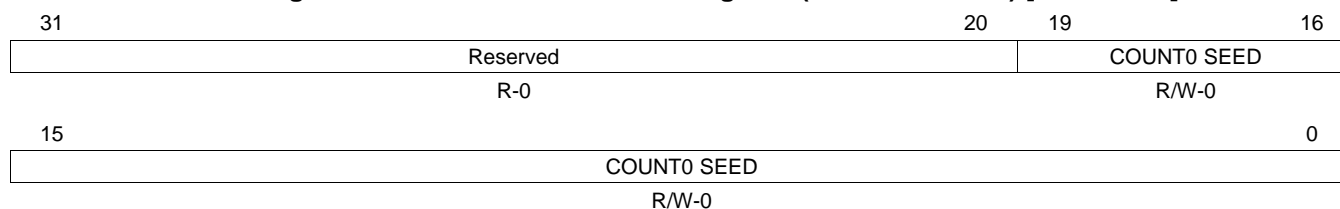
Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-12	DONE INT ENA	5h Others	Done Interrupt Enable. Any operation mode read, privileged mode write: No interrupt is generated when the DONE flag is set in the DCC Status (DCCSTAT) register. DONE interrupt is generated when the DONE flag is set in the DCC Status (DCCSTAT) register.
11-8	SINGLE SHOT	Ah Bh other	Single-Shot Mode Enable. Any operation mode read, privileged mode write: DCC stops counting when counter0 and valid0 both reach zero. DCC stops counting when counter1 reaches zero. DCC counts continuously and only stops when an error occurs.
7-4	ERR ENA	5h Others	Error Interrupt Enable. Any operation mode read, privileged mode write: No interrupt is generated when the ERR flag is set in the DCC Status (DCCSTAT) register. ERROR interrupt is generated when the ERR flag is set in the DCC Status (DCCSTAT) register.
3-0	DCC ENA	5h Others	DCC Enable. Any operation mode read, privileged mode write: All DCC counters are stopped and error-checking is disabled. Read: Counters are enabled. Write: Load counters with their seed values and begin counting. It is recommended to write 1010 to enable counters to protect against single-bit errors.



### 11.4.3 DCC Counter0 Seed Register (DCCNT0SEED)

Figure 11-9 and Table 11-8 describe the DCC Counter0 Seed register.

**Figure 11-9. DCC Counter0 Seed Register (DCCNT0SEED) [offset = 8h]**



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

**Table 11-8. DCC Counter0 Seed Register (DCCNT0SEED) Field Descriptions**

Bit	Field	Value	Description
31-20	Reserved	0	Read returns 0. Writes have no effect.
19-0	COUNT0 SEED		Seed value for DCC Counter0. Read in any operating mode returns the current value of Counter0. Writing in privileged mode only sets the current seed value for Counter0.

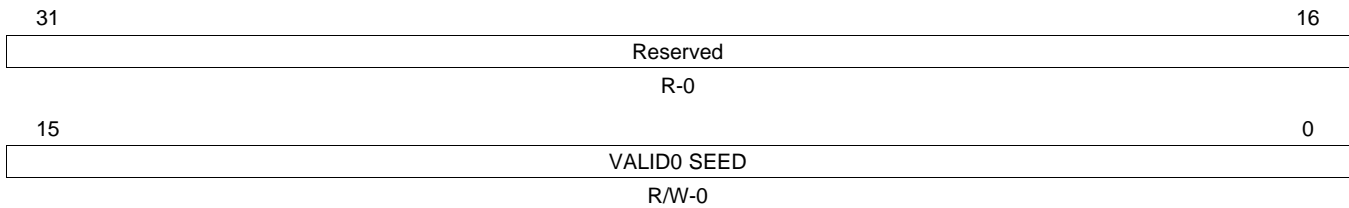
**NOTE: Seed for Counter0 must be non-zero**

The DCC must only be enabled after programming a non-zero value in the COUNT0 SEED register.

### 11.4.4 DCC Valid0 Seed Register (DCCVALID0SEED)

Figure 11-10 and Table 11-9 describe the DCC Valid0 Seed register.

**Figure 11-10. DCC Valid0 Seed Register (DCCVALID0SEED) [offset = Ch]**



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

**Table 11-9. DCC Valid0 Seed Register (DCCVALID0SEED) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	VALID0 SEED		Seed value for DCC Valid0. This value defines the window within which the Counter1 must reach 0. This window needs to be at least 4 cycles wide. Read in any operating mode returns the current value of Valid0. Writing in privileged mode only sets the current seed value for Valid0.

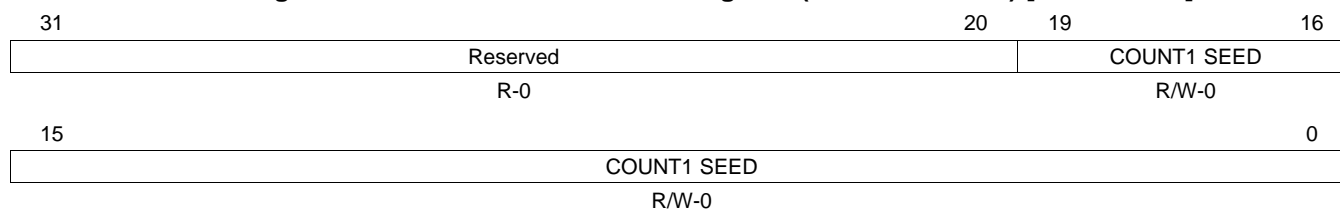
**NOTE: Seed for Valid0 must be at least 0x4**

The DCC must only be enabled after programming a value greater than or equal to 0x4 in the VALID0 SEED register.

### 11.4.5 DCC Counter1 Seed Register (DCCNT1SEED)

Figure 11-11 and Table 11-10 describe the DCC Counter1 Seed register.

**Figure 11-11. DCC Counter1 Seed Register (DCCNT1SEED) [offset = 10h]**



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

**Table 11-10. DCC Counter1 Seed Register (DCCNT0SEED) Field Descriptions**

Bit	Field	Value	Description
31-20	Reserved	0	Read returns 0. Writes have no effect.
19-0	COUNT1 SEED		Seed value for DCC Counter1. Read in any operating mode returns the current value of Counter1. Writing in privileged mode only sets the current seed value for Counter1.

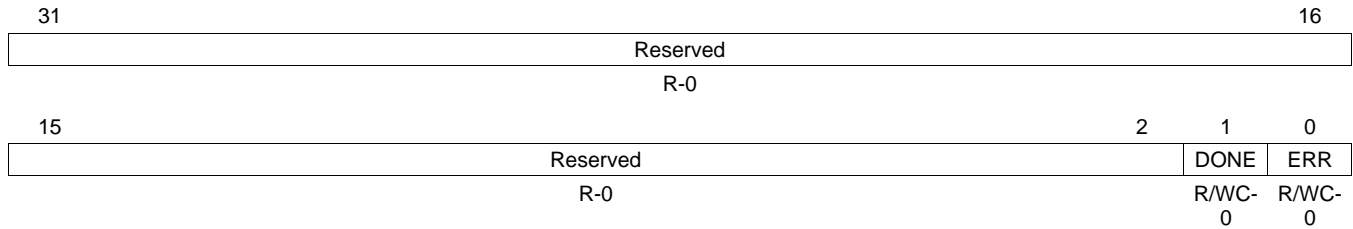
**NOTE: Seed for Counter0 must be non-zero**

The DCC must only be enabled after programming a non-zero value in the COUNT1 SEED register.

### 11.4.6 DCC Status Register (DCCSTAT)

Figure 11-7 and Table 11-6 describe the DCC Status register.

**Figure 11-12. DCC Status Register (DCCSTAT) [offset = 14h]**



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

**Table 11-11. DCC Status Register (DCCSTAT) Field Descriptions**

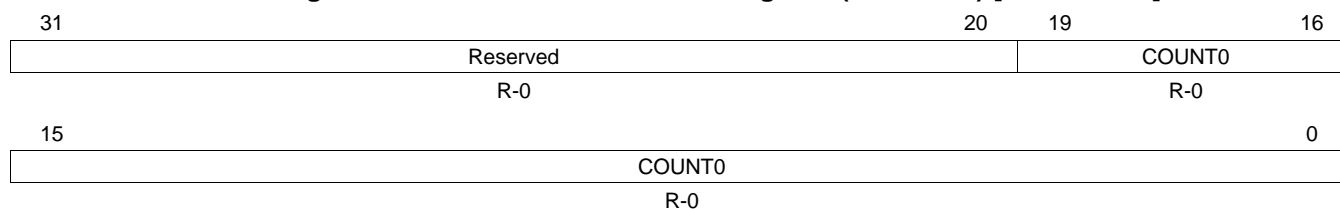
Bit	Field	Value	Description
31-2	Reserved	0	Read returns 0. Writes have no effect.
1	DONE FLG	0	Single-Shot Sequence Done flag. Indicates that a single-shot DCC sequence is done without any error. Read: Single-shot sequence is not done. Write: Writing 0 has no effect.
		1	Read: Single-shot sequence is done without any error. Write: Writing 1 in privileged mode clears the DONE FLG.
0	ERR FLG	0	Error flag. Indicates that a DCC error has occurred. Read: DCC error has not occurred. Write: Writing 0 has no effect.
		1	Read: An error has occurred. Write: Writing 1 in privileged mode clears the ERR FLG.



### 11.4.7 DCC Counter0 Value Register (DCCCNT0)

Figure 11-13 and Table 11-12 describe the DCC Counter0 Value register.

**Figure 11-13. DCC Counter0 Value Register (DCCCNT0) [offset = 18h]**



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

**Table 11-12. DCC Counter0 Seed Register (DCCCNT0SEED) Field Descriptions**

Bit	Field	Value	Description
31-20	Reserved	0	Read returns 0. Writes have no effect.
19-0	COUNT0		Current value of DCC Counter0. Read in any operating mode returns the current value of Counter0. Writes have no effect.

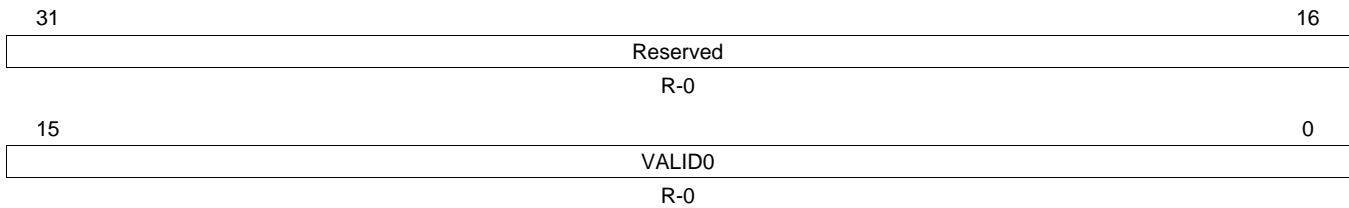
**NOTE: Reads may not return exact current value of counter**

Reading the counter0 value while counting is enabled may not return the exact value of the counter0.

### 11.4.8 DCC Valid0 Value Register (DCCVALID0)

Figure 11-14 and Table 11-13 describe the DCC Valid0 Value register.

**Figure 11-14. DCC Valid0 Value Register (DCCVALID0) [offset = 1Ch]**



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

**Table 11-13. DCC Valid0 Value Register (DCCVALID0) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	VALID0		Current value for DCC Valid0. Read in any operating mode returns the current value of Valid0. Writes have no effect.

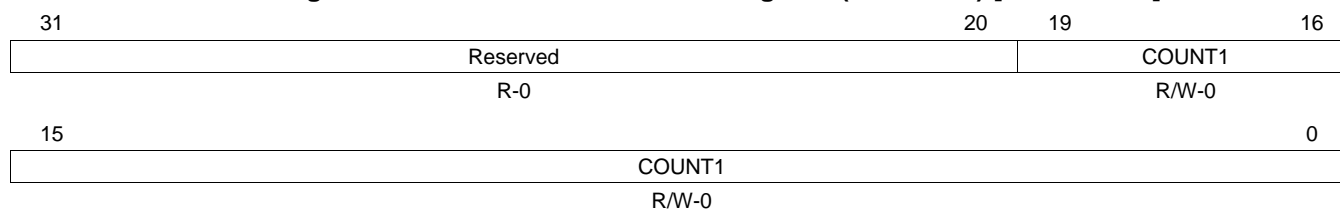
**NOTE: Reads may not return exact current value of Valid0**

Reading the Valid0 value while counting is enabled may not return the exact value of the Valid0.

### 11.4.9 DCC Counter1 Value Register (DCCCNT1)

Figure 11-15 and Table 11-14 describe the DCC Counter1 Value register.

**Figure 11-15. DCC Counter1 Value Register (DCCCNT1) [offset = 20h]**



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

**Table 11-14. DCC Counter1 Value Register (DCCCNT1) Field Descriptions**

Bit	Field	Value	Description
31-20	Reserved	0	Read returns 0. Writes have no effect.
19-0	COUNT1		Current value for DCC Counter1. Read in any operating mode returns the current value of Counter1. Writes have no effect

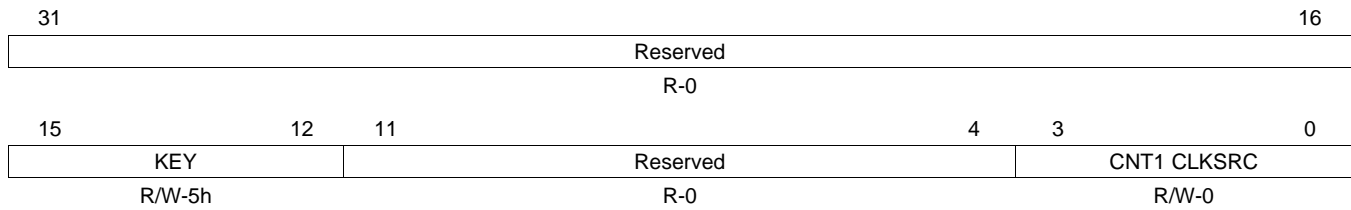
**NOTE: Reads may not return exact current value of counter**

Reading the counter1 value while counting is enabled may not return the exact value of the counter1.

### 11.4.10 DCC Counter1 Clock Source Selection Register (DCCNT1CLKSRC)

Figure 11-15 and Table 11-14 describe the DCC Counter1 Value register.

**Figure 11-16. DCC Counter1 Clock Source Selection Register (DCCNT1CLKSRC) [offset = 24h]**



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

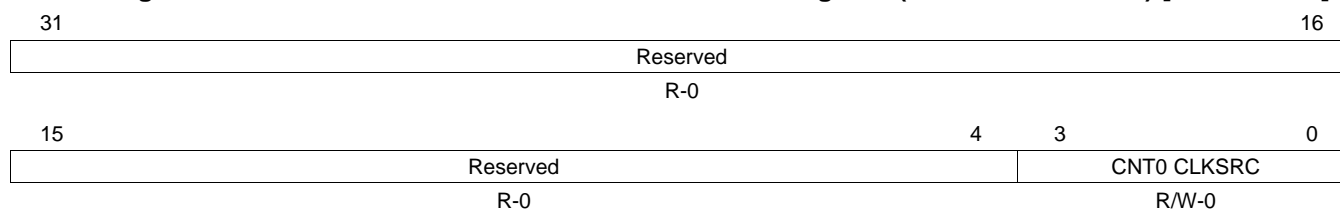
**Table 11-15. DCC Counter1 Value Register (DCCNT1) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-12	KEY		Key to enable clock source selection for Counter1. Read in any operating mode returns the current value of the key. Writing in privileged mode sets the key value. Writing 1010 as the key enables the CNT1 CLKSRC field to define the clock source for counter1. Writing any other key value disables the clock source selection for counter1. In this case, the N2HET1[31] signal is used as the source for Counter1 for DCC1, and the N2HET2[0] signal is used as the source for Counter1 for DCC2. See <a href="#">Table 11-2</a> and <a href="#">Table 11-4</a> .
11-4	Reserved	0	Read returns 0. Writes have no effect.
3-0	CNT1 CLKSRC		Clock Source for Counter1 when KEY is programmed to be 0xA. Read in any operating mode returns the current value of CLKSRC. Writes in privileged mode select the clock source for Counter1. See <a href="#">Table 11-2</a> and <a href="#">Table 11-4</a> .

### 11.4.11 DCC Counter0 Clock Source Selection Register (DCCNT0CLKSRC)

Figure 11-15 and Table 11-14 describe the DCC Counter1 Value register.

**Figure 11-17. DCC Counter1 Clock Source Selection Register (DCCNT1CLKSRC) [offset = 24h]**



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

**Table 11-16. DCC Counter1 Value Register (DCCNT1) Field Descriptions**

Bit	Field	Value	Description
31-4	Reserved	0	Read returns 0. Writes have no effect.
3-0	CNT0 CLKSRC		Clock Source for Counter0 Read in any operating mode returns the current value of CLKSRC. Writes in privileged mode select the clock source for Counter1. See <a href="#">Table 11-1</a> and <a href="#">Table 11-3</a> .

## **Error Signaling Module (ESM)**

---

---

This chapter provides the details of the error signaling module (ESM), which aggregates device errors and provides internal and external error response based on error severity.

<b>Topic</b>	<b>Page</b>
<b>12.1 Overview</b> .....	<b>428</b>
<b>12.2 Module Operation</b> .....	<b>430</b>
<b>12.3 Recommended Programming Procedure</b> .....	<b>433</b>
<b>12.4 Control Registers</b> .....	<b>434</b>

## 12.1 Overview

The Error Signaling Module (ESM) collects and reports the various error conditions on the TMS570 microcontroller. The error condition is categorized based on a severity level. Error response is then generated based on the category of the error. Possible error responses include a low priority interrupt, high priority interrupt, and an external pin action.

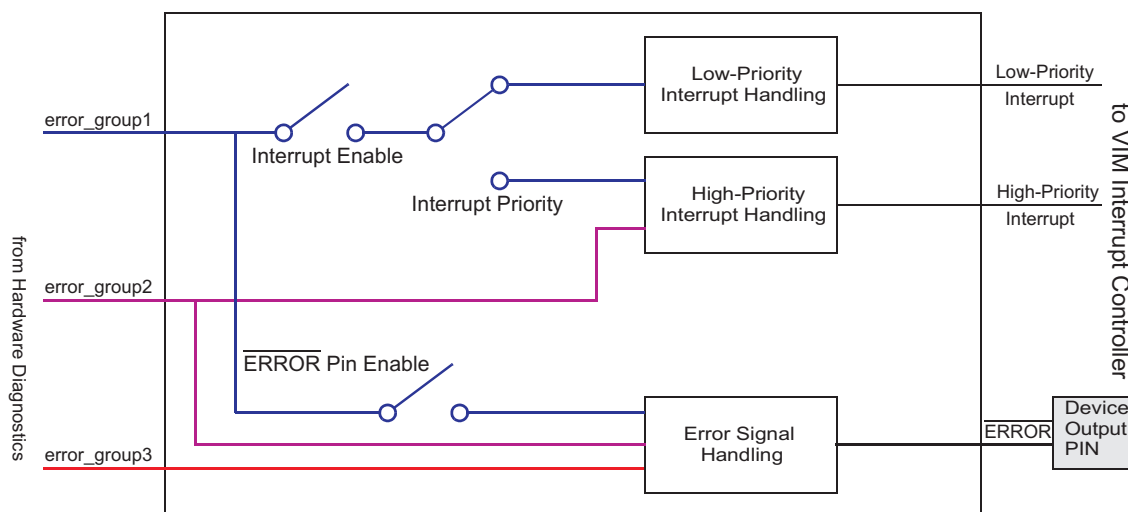
### 12.1.1 Feature List

- Up to 128 error channels are supported, divided into 3 different groups:
  - 64 Group1 (low severity) channels with configurable interrupt generation and configurable  $\overline{\text{ERROR}}$  pin behavior
  - 32 Group2 (high severity) channels with predefined interrupt generation and predefined  $\overline{\text{ERROR}}$  pin behavior
  - 32 Group3 (high severity) channels with no interrupt generation and predefined  $\overline{\text{ERROR}}$  pin behavior. These channels have no interrupt response as they are reserved for CPU based diagnostics which generate aborts directly to the CPU.
- Dedicated device  $\overline{\text{ERROR}}$  pin to signal an external observer
- Configurable timebase for  $\overline{\text{ERROR}}$  pin output
- Error forcing capability for latent fault testing

### 12.1.2 Block Diagram

As shown in Figure 12-1, the ESM channels are divided into three groups. Group1 channels are considered to be low severity. Group1 errors have a configurable interrupt response and configurable  $\overline{\text{ERROR}}$  pin behavior. Group2 channels are  $\overline{\text{ERROR}}$  high severity. Group2 errors always generate a high priority interrupt and an output on the  $\overline{\text{ERROR}}$  pin. Group3 errors are reserved for high severity errors generated by diagnostics which have already generated a CPU abort response. Because an abort response is generated, there is no need to generate an interrupt response. Group3 errors always generate an  $\overline{\text{ERROR}}$  pin output.

**Figure 12-1. Block Diagram**



The ESM interrupt and  $\overline{\text{ERROR}}$  pin behavior are also summarized in Table 12-1 below:

**Table 12-1. ESM interrupt and  $\overline{\text{ERROR}}$  pin behavior**

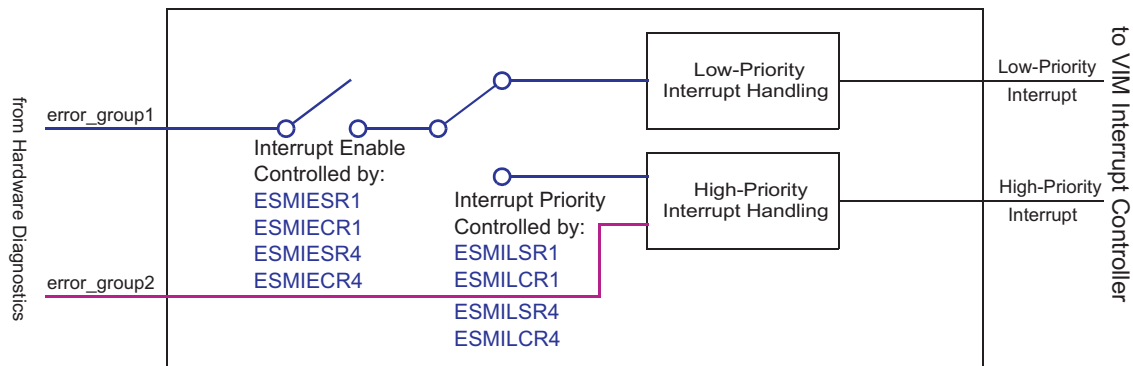
Error Group	Interrupt Generated?	Interrupt priority?	$\overline{\text{ERROR}}$ pin response generated?
1	configurable interrupt	configurable priority	configurable output generation
2	interrupt generated	high priority	output generated

**Table 12-1. ESM interrupt and  $\overline{\text{ERROR}}$  pin behavior (continued)**

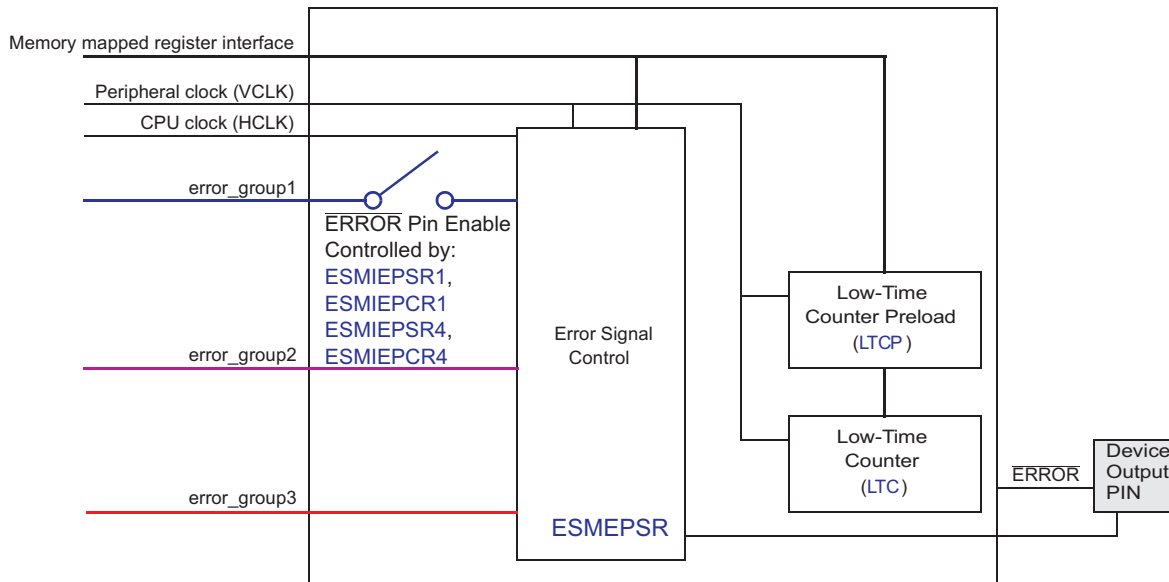
Error Group	Interrupt Generated?	Interrupt priority?	$\overline{\text{ERROR}}$ pin response generated?
3	no interrupt	NA	output generated

Figure 12-2 and Figure 12-3 shows the interrupt response handling and  $\overline{\text{ERROR}}$  pin response handling with register configuration. The total active time of the  $\overline{\text{ERROR}}$  pin is controlled by the Low-Time Counter Preload register (LTCP) and key register (ESMEPSR) as shown in Figure 12-3. Please see Section 12.2.2 for details.

**Figure 12-2. Interrupt Response Handling**



**Figure 12-3.  $\overline{\text{ERROR}}$  Pin Response Handling**





## 12.2 Module Operation

This device has 128 error channels, divided into 3 different error groups. Please refer to the device datasheet for ESM channel assignment details.

The ESM module has error flags for each error channel. The error status registers ESMSR1, ESMSR4, ESMSR2, ESMSR3 provide status information on a pending error of Group1 (Channel 0-31), Group1 (Channel 32-63), Group2, and Group3, respectively. Register (ESMEPSR) provides the current  $\overline{\text{ERROR}}$  status. The module also provides a status shadow register ESMSR2, which maintains the error flags of Group2 until power-on reset ( $\overline{\text{PORRST}}$ ) is asserted. Please see [Section 12.2.1](#) for details of their behavior during power on reset and warm reset.

Once an error occurs, the ESM module will set the corresponding error flags. In addition, it can trigger an interrupt,  $\overline{\text{ERROR}}$  pin outputs low depending on the ESM settings. Once the  $\overline{\text{ERROR}}$  pin outputs low, a power on reset or a write of 0x5 to ESMEKR is required to release the ESM back to normal state. Please see [Section 12.2.2](#) for details. The application can read the error status registers (ESMSR1, ESMSR4, ESMSR2, and ESMSR3) to debug the error. If an  $\overline{\text{RST}}$  is triggered or the error interrupt has been served, the error flag of Group2 should be read from ESMSR2 because the error flag in ESMSR2 will be cleared by  $\overline{\text{RST}}$ .

The user can also test the functionality of the  $\overline{\text{ERROR}}$  pin by forcing an error. Please see [Section 12.2.3](#) for details.

### 12.2.1 Reset Behavior

Power on reset:

- $\overline{\text{ERROR}}$  pin behavior

When nPORRST is active, the  $\overline{\text{ERROR}}$  pin is in a high impedance state (output drivers disabled).

- Register behavior

After  $\overline{\text{PORRST}}$ , all registers in ESM module will be re-initialized to the default value. All the error status registers are cleared to zero.

Warm reset ( $\overline{\text{RST}}$ ):

- $\overline{\text{ERROR}}$  pin behavior

During  $\overline{\text{RST}}$ , the  $\overline{\text{ERROR}}$  pin is in “output active” state with pull-down disabled. The  $\overline{\text{ERROR}}$  pin remains unchanged after  $\overline{\text{RST}}$ .

- Register behavior

After  $\overline{\text{RST}}$ , ESMSR1, ESMSR4, ESMSR2, ESMSR3 and ESMEPSR register values remains un-changed. Since  $\overline{\text{RST}}$  does not clear the critical failure registers, the user can read those registers to debug the failures even if an  $\overline{\text{RST}}$  is triggered.

After  $\overline{\text{RST}}$ , if one of the flags in ESMSR1 and ESMSR4 is set, the interrupt service routine will be called once the corresponding interrupt is enabled.

---

**NOTE:** ESMSR2 is cleared after  $\overline{\text{RST}}$ . The flag in ESMSR2 gets cleared when reading the appropriate vector in the ESMIOFFHR offset register. Reading ESMIOFFHR will not clear the ESMSR1, ESMSR4 and the shadow register ESMSR2. Reading ESMIOFFLR will also not clear the ESMSR1 and ESMSR4.

---

### 12.2.2 $\overline{\text{ERROR}}$ Pin Timing

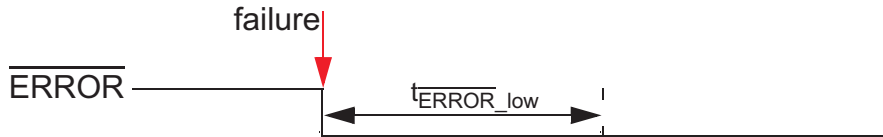
The  $\overline{\text{ERROR}}$  pin is an active low function. The state of the pin is also readable from  $\overline{\text{ERROR}}$  Pin Status Register (ESMEPSR). A warm reset ( $\overline{\text{RST}}$ ) does not affect the state of the pin. The pin is in a high-impedance state during power-on reset. Once the ESM module drives the  $\overline{\text{ERROR}}$  pin low, it remains in this state for the time specified by the Low-Time Counter Preload register (LTCPR). Based on the time period of the peripheral clock (VCLK), the total active time of the  $\overline{\text{ERROR}}$  pin can be calculated as:

$$t_{\overline{\text{ERROR}}\_low} = t_{VCLK} \times (\text{LTCPR} + 1) \quad (22)$$

Once this period expires, the  $\overline{\text{ERROR}}$  pin is set to high in case the reset of the  $\overline{\text{ERROR}}$  pin was requested. This request is done by writing an appropriate key (0x5) to the key register (ESMEKR) during the  $\overline{\text{ERROR}}$  pin low time. Here are a few examples:

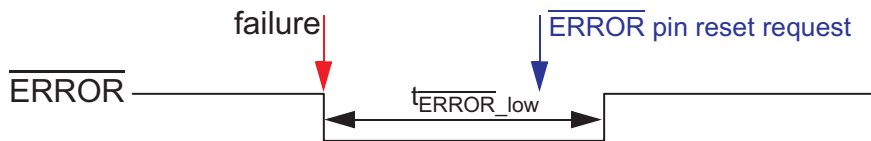
Example 1: ESM detects a failure and drives the  $\overline{\text{ERROR}}$  pin low. No  $\overline{\text{ERROR}}$  pin reset is requested. The  $\overline{\text{ERROR}}$  pin continues outputting low until power on reset occurs.

Figure 12-4.  $\overline{\text{ERROR}}$  Pin Timing - Example 1



Example 2: ESM detects a failure and drives the  $\overline{\text{ERROR}}$  pin low. An  $\overline{\text{ERROR}}$  pin reset request is received before  $t_{\text{ERROR\_low}}$  expires. In this case, the  $\overline{\text{ERROR}}$  pin is set to high immediately after  $t_{\text{ERROR\_low}}$  expires.

Figure 12-5.  $\overline{\text{ERROR}}$  Pin Timing - Example 2



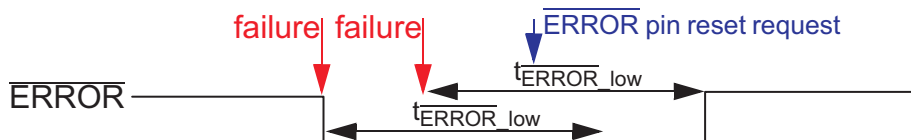
Example 3: ESM detects a failure and drives the  $\overline{\text{ERROR}}$  pin low. An  $\overline{\text{ERROR}}$  pin reset request is received after  $t_{\text{ERROR\_low}}$  expires. In this case, the  $\overline{\text{ERROR}}$  pin is set to high immediately after  $\overline{\text{ERROR}}$  pin reset request is received.

Figure 12-6.  $\overline{\text{ERROR}}$  Pin Timing - Example 3



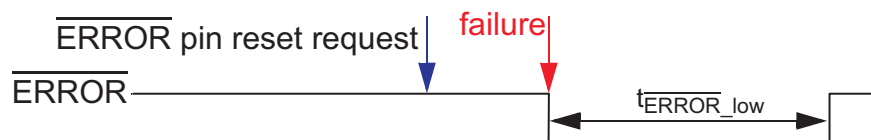
Example 4: ESM detects a failure and drives the  $\overline{\text{ERROR}}$  pin low. Another failure occurs within the time the pin stays low. In this case, the low time counter will be reset when the other failure occurs. In other words,  $t_{\text{ERROR\_low}}$  should be counted from whenever the most recent failure occurs.

Figure 12-7.  $\overline{\text{ERROR}}$  Pin Timing - Example 4



Example 5: The reset of the  $\overline{\text{ERROR}}$  pin was requested by the software even before the failure occurs. In this case, the  $\overline{\text{ERROR}}$  pin is set to high immediately after  $t_{\text{ERROR\_low}}$  expires. This case is not recommended and should be avoided by the application.

Figure 12-8.  $\overline{\text{ERROR}}$  Pin Timing - Example 5



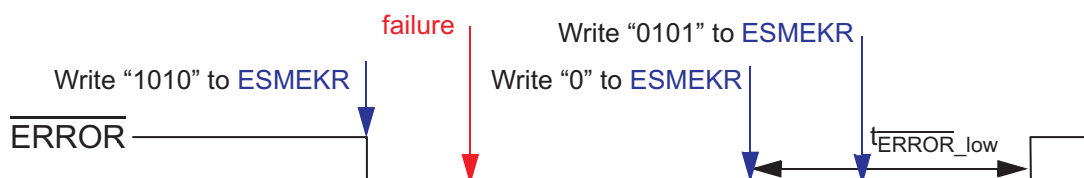
### 12.2.3 Forcing an Error Condition

The error response generation mechanism is testable by software by forcing an error condition. This allows testing the  $\overline{\text{ERROR}}$  pin functionality. By writing a dedicated key to the error forcing key register (ESMEKR), the  $\overline{\text{ERROR}}$  pin is set to low for the specified time. The following steps describe how to force an error condition:

1. Check  $\overline{\text{ERROR}}$  Pin Status Register (ESMEPSR). This register must be “1” to switch into the error forcing mode.  
The ESM module cannot be switched into the error forcing mode if a failure has already been detected in functional mode. The application command to switch to error forcing mode is ignored.
2. Write “1010b” to the error forcing key register (ESMEKR). After that, the  $\overline{\text{ERROR}}$  pin should output low (error force mode).

Once the application puts the ESM module in the error forcing mode, the  $\overline{\text{ERROR}}$  pin cannot indicate the normal error functionality. If a failure occurs during this time, it gets still latched and the LTC is reset and stopped. The error output pin is already driven low on account of the error forcing mode. When the ESM is forced back to normal functional mode the LTC becomes active and forces the  $\overline{\text{ERROR}}$  pin low until the expiration of the LTC (see [Figure 12-9](#)).

**Figure 12-9. ERROR Pin Timing - Example 6**

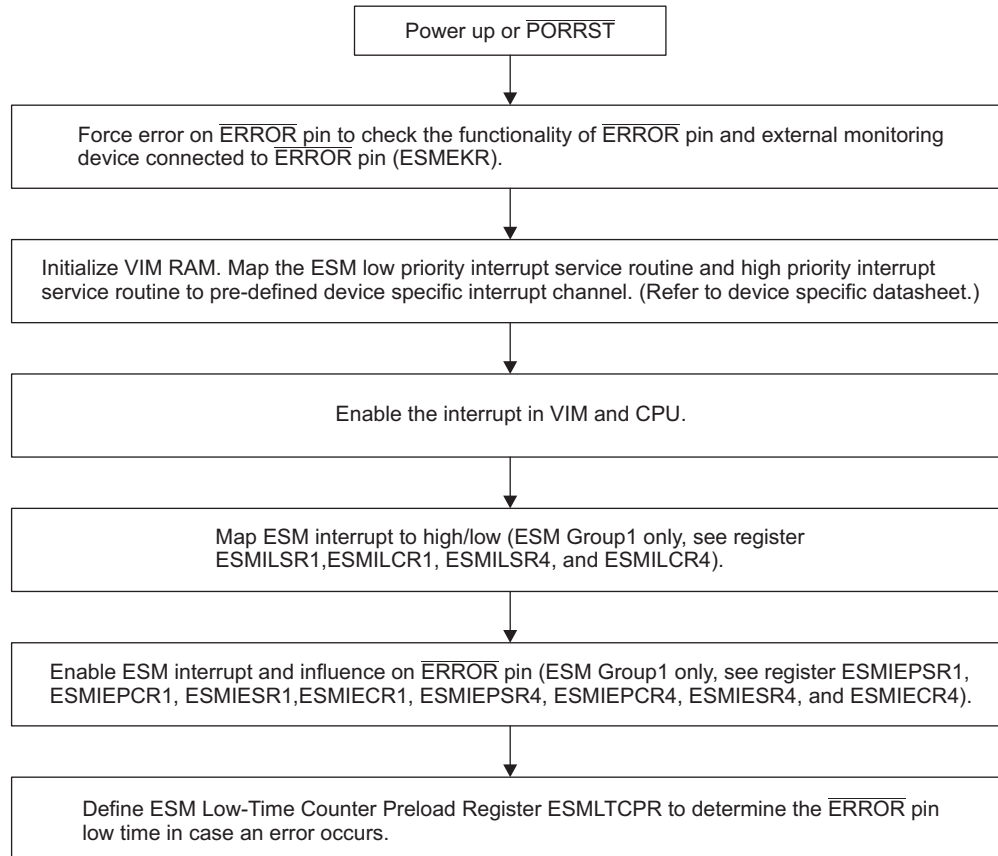


3. Write “0000” to the error forcing key register (ESMEKR) back to the active normal mode.  
If there are no errors detected while the ESM module is in the error forcing mode, the  $\overline{\text{ERROR}}$  pin goes high immediately after exiting the error forcing mode.

## 12.3 Recommended Programming Procedure

During the initialization stage, the application code should follow the recommendations in [Figure 12-10](#) to initialize the ESM.

**Figure 12-10. ESM Initialization**



Once an error occurs, it can trigger an interrupt,  $\overline{\text{ERROR}}$  pin outputs low depending on the ESM settings. Once the  $\overline{\text{ERROR}}$  pin outputs low, a power on reset or a write of 0x5 to ESMEKR is required to release the ESM back to normal state. The application can read the error status registers (ESMSR1, ESMSR4, ESMSR2, and ESMSR3) to debug the error. If an  $\overline{\text{RST}}$  is triggered or the error interrupt has been served, the error flag of Group2 should be read from ESMSR2 because the error flag in ESMSR2 will be cleared by  $\overline{\text{RST}}$ .

## 12.4 Control Registers

This section describes the ESM registers. Each register begins on a 32-bit word boundary. The registers support 32-bit, 16-bit and 8-bit accesses. The base address for the control registers is 0xFFFF500.

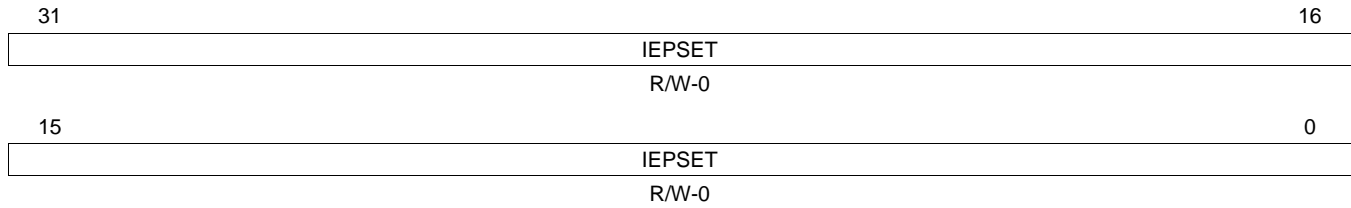
**Table 12-2. ESM Module Registers**

Offset	Acronym	Register Description	Section
FFFF F500h	ESMEEPAPR1	ESM Enable <b>ERROR</b> Pin Action/Reponse Register 1	<a href="#">Section 12.4.1</a>
FFFF F504h	ESMDEPAPR1	ESM Disable <b>ERROR</b> Pin Action/Reponse Register 1	<a href="#">Section 12.4.2</a>
FFFF F508h	ESMIESR1	ESM Interrupt Enable Set/Status Register 1	<a href="#">Section 12.4.3</a>
FFFF F50Ch	ESMIECR1	ESM Interrupt Enable Clear/Status Register 1	<a href="#">Section 12.4.4</a>
FFFF F510h	ESMILSR1	Interrupt Level Set/Status Register 1	<a href="#">Section 12.4.5</a>
FFFF F514h	ESMILCR1	Interrupt Level Clear/Status Register 1	<a href="#">Section 12.4.6</a>
FFFF F518h	ESMSR1	ESM Status Register 1	<a href="#">Section 12.4.7</a>
FFFF F51Ch	ESMSR2	ESM Status Register 2	<a href="#">Section 12.4.8</a>
FFFF F520h	ESMSR3	ESM Status Register 3	<a href="#">Section 12.4.9</a>
FFFF F524h	ESMEPSR	ESM <b>ERROR</b> Pin Status Register	<a href="#">Section 12.4.10</a>
FFFF F528h	ESMIOFFHR	ESM Interrupt Offset High Register	<a href="#">Section 12.4.11</a>
FFFF F52Ch	ESMIOFFLR	ESM Interrupt Offset Low Register	<a href="#">Section 12.4.12</a>
FFFF F530h	ESMLTCR	ESM Low-Time Counter Register	<a href="#">Section 12.4.13</a>
FFFF F534h	ESMLTCPR	ESM Low-Time Counter Preload Register	<a href="#">Section 12.4.14</a>
FFFF F538h	ESMEKR	ESM Error Key Register	<a href="#">Section 12.4.15</a>
FFFF F53Ch	ESMSSR2	ESM Status Shadow Register 2	<a href="#">Section 12.4.16</a>
FFFF F540h	ESMIEPSR4	ESM Influence <b>ERROR</b> Pin Set/Status Register 4	<a href="#">Section 12.4.17</a>
FFFF F544h	ESMIEPCR4	ESM Influence <b>ERROR</b> Pin Clear/Status Register 4	<a href="#">Section 12.4.18</a>
FFFF F548h	ESMIESR4	ESM Interrupt Enable Set/Status Register 4	<a href="#">Section 12.4.19</a>
FFFF F54Ch	ESMIECR4	ESM Interrupt Enable Clear/Status Register 4	<a href="#">Section 12.4.20</a>
FFFF F550h	ESMILSR4	Interrupt Level Set/Status Register 4	<a href="#">Section 12.4.21</a>
FFFF F554h	ESMILCR4	Interrupt Level Clear/Status Register 4	<a href="#">Section 12.4.22</a>
FFFF F558h	ESMSR4	ESM Status Register 4	<a href="#">Section 12.4.23</a>

**12.4.1 ESM Enable  $\overline{\text{ERROR}}$  Pin Action/Response Register 1 (ESMEEPAPR1)**

The base address for this register is FFFF F500h. This register is dedicated for Group1.

**Figure 12-11. ESM Enable  $\overline{\text{ERROR}}$  Pin Action/Response Register 1 (ESMEEPAPR1)**



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

**Table 12-3. ESM Enable  $\overline{\text{ERROR}}$  Pin Action/Response Register 1 (ESMEEPAPR1) Field Descriptions**

Bit	Field	Value	Description
31-0	IEPSET	0	Enable $\overline{\text{ERROR}}$ Pin Action/Response on Group 1. <b>User and privileged mode:</b> Read: Failure on channel x has no influence on $\overline{\text{ERROR}}$ pin. <b>Privileged mode:</b> Write: Leaves the bit and the corresponding clear bit in the ESMIEPCR1 register unchanged.
		1	<b>User and privileged mode:</b> Read: Failure on channel x has influence on $\overline{\text{ERROR}}$ pin. <b>Privileged mode:</b> Write: Enables failure influence on $\overline{\text{ERROR}}$ pin and sets the corresponding clear bit in the ESMIEPCR1 register.

### 12.4.2 ESM Disable $\overline{\text{ERROR}}$ Pin Action/Response Register 1 (ESMDEPAPR1)

The base address for this register is FFFF F504h. This register is dedicated for Group1.

**Figure 12-12. ESM Disable  $\overline{\text{ERROR}}$  Pin Action/Response Register 1 (ESMDEPAPR1)**

31	IEPCLR	16
	R/W-0	
15	IEPCLR	0
	R/W-0	

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

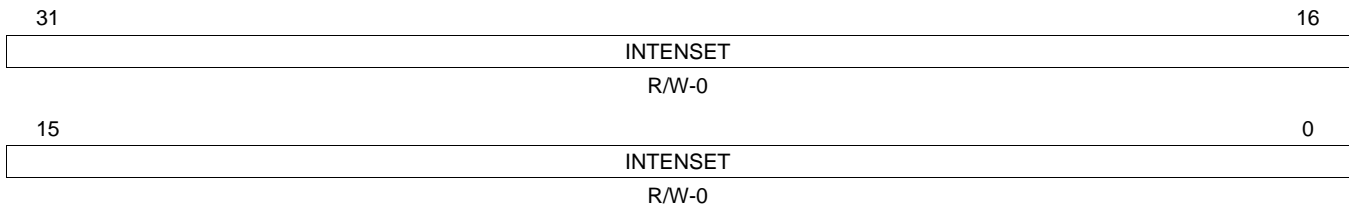
**Table 12-4. ESM Disable  $\overline{\text{ERROR}}$  Pin Action/Response Register 1 (ESMDEPAPR1) Field Descriptions**

Bit	Field	Value	Description
31-0	IEPCLR	0	Disable $\overline{\text{ERROR}}$ Pin Action/Response on Group 1. <b>User and privileged mode:</b> Read: Failure on channel x has no influence on $\overline{\text{ERROR}}$ pin. <b>Privileged mode:</b> Write: Leaves the bit and the corresponding set bit in the ESMIEPSR1 register unchanged.
		1	<b>User and privileged mode:</b> Read: Failure on channel x has influence on $\overline{\text{ERROR}}$ pin. <b>Privileged mode:</b> Write: Disables failure influence on $\overline{\text{ERROR}}$ pin and clears the corresponding set bit in the ESMIEPSR1 register.

### 12.4.3 ESM Interrupt Enable Set/Status Register 1 (ESMIESR1)

The base address for this register is FFFF F508h. This register is dedicated for Group1.

**Figure 12-13. ESM Interrupt Enable Set/Status Register 1 (ESMIESR1)**



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

**Table 12-5. ESM Interrupt Enable Set/Status Register 1 (ESMIESR1) Field Descriptions**

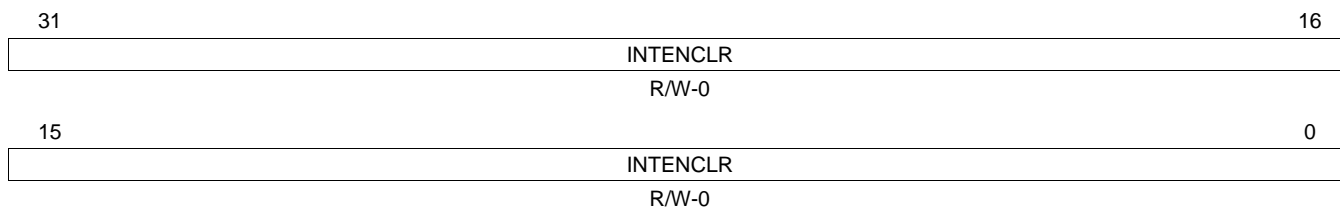
Bit	Field	Value	Description
31-0	INTENSET	0	Set interrupt Enable <b>User and privileged mode:</b> Read: Interrupt is disabled.
		1	<b>Privileged mode:</b> Write: Leaves the bit and the corresponding clear bit in the ESMIECR1 register unchanged. <b>User and privileged mode:</b> Read: Interrupt is enabled. <b>Privileged mode:</b> Write: Enables interrupt and sets the corresponding clear bit in the ESMIECR1 register.



### 12.4.4 ESM Interrupt Enable Clear/Status Register 1 (ESMIECR1)

The base address for this register is FFFF F50Ch. This register is dedicated for Group1.

**Figure 12-14. ESM Interrupt Enable Clear/Status Register 1 (ESMIECR1)**



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

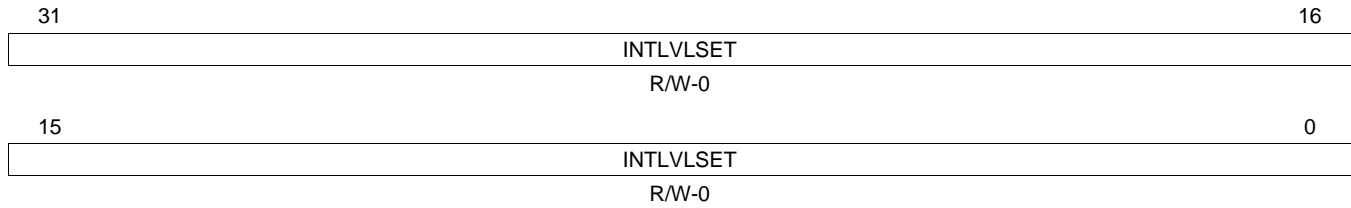
**Table 12-6. ESM Interrupt Enable Clear/Status Register 1 (ESMIECR1) Field Descriptions**

Bit	Field	Value	Description
31-0	INTENCLR	0	Clear Interrupt Enable <b>User and privileged mode:</b> Read: Interrupt is disabled. <b>Privileged mode:</b> Write: Leaves the bit and the corresponding set bit in the ESMIESR1 register unchanged.
		1	<b>User and privileged mode:</b> Read: Interrupt is enabled. <b>Privileged mode:</b> Write: Disables interrupt and clears the corresponding set bit in the ESMIESR1 register.

### 12.4.5 ESM Interrupt Level Set/Status Register 1 (ESMILSR1)

The base address for this register is FFFF F510h. This register is dedicated for Group1.

**Figure 12-15. ESM Interrupt Level Set/Status Register 1 (ESMILSR1)**



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

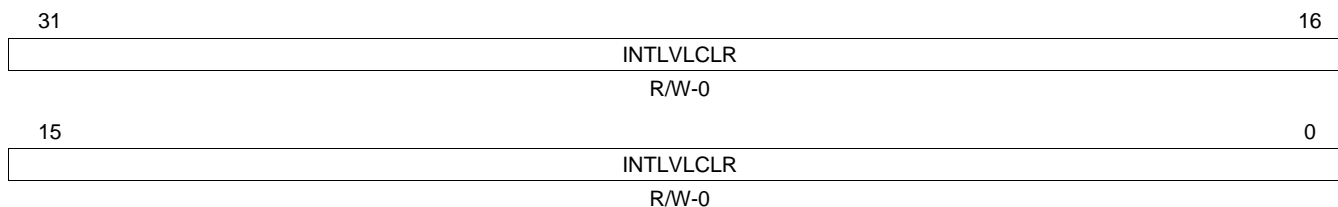
**Table 12-7. ESM Interrupt Level Set/Status Register 1 (ESMILSR1) Field Descriptions**

Bit	Field	Value	Description
31-0	INTLVLSET	0	Set Interrupt Priority <b>User and privileged mode:</b> Read: Interrupt of channel x is mapped to low level interrupt line. <b>Privileged mode:</b> Write: Leaves the bit and the corresponding clear bit in the ESMILCR1 register unchanged.
		1	<b>User and privileged mode:</b> Read: Interrupt of channel x is mapped to high level interrupt line. <b>Privileged mode:</b> Write: Maps interrupt of channel x to high level interrupt line and sets the corresponding clear bit in the ESMILCR1 registe.

### 12.4.6 ESM Interrupt Level Clear/Status Register 1 (ESMILCR1)

The base address for this register is FFFF F514h. This register is dedicated for Group1.

**Figure 12-16. ESM Interrupt Level Clear/Status Register 1 (ESMILCR1)**



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

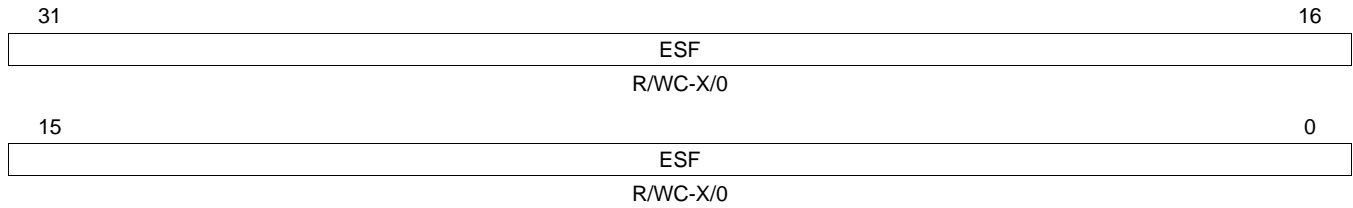
**Table 12-8. ESM Interrupt Level Clear/Status Register 1 (ESMILCR1) Field Descriptions**

Bit	Field	Value	Description
31-0	INTLVLCLR	0	Clear Interrupt Priority. <b>User and privileged mode:</b> Read: Interrupt of channel x is mapped to low level interrupt line. <b>Privileged mode:</b> Write: Leaves the bit and the corresponding set bit in the ESMILSR1 register unchanged.
		1	<b>User and privileged mode:</b> Read: Interrupt of channel x is mapped to high level interrupt line. <b>Privileged mode:</b> Write: Maps interrupt of channel x to low level interrupt line and clears the corresponding set bit in the ESMILSR1 register.

### 12.4.7 ESM Status Register 1 (ESMSR1)

The base address for this register is FFFF F518h. This register is dedicated for Group1.

**Figure 12-17. ESM Status Register 1 (ESMSR1)**



LEGEND: R = Read; W = Write; C = Clear; -n = value after reset/PORRST; X = Value unchanged

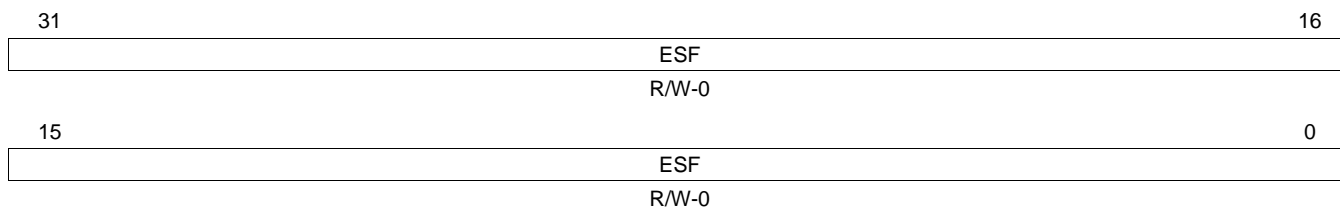
**Table 12-9. ESM Status Register 1 (ESMSR1) Field Descriptions**

Bit	Field	Value	Description
31-0	ESF	0	<p>Error Status Flag. Provides status information on a pending error.</p> <p><b>User and privileged mode:</b> Read: No error occurred; no interrupt is pending</p> <p><b>Privileged mode:</b> Write: Leaves the bit unchanged.</p>
		1	<p><b>User and privileged mode:</b> Read: Error occurred; interrupt is pending.</p> <p><b>Privileged mode:</b> Write: Clears the bit.</p> <p><b>Note:</b> After nRST, if one of these flags are set and the corresponding interrupt are enabled, the interrupt service routine will be called.</p>

### 12.4.8 ESM Status Register 2 (ESMSR2)

The base address for this register is FFFF F51Ch. This register is dedicated for Group2.

**Figure 12-18. ESM Status Register 2 (ESMSR2)**



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

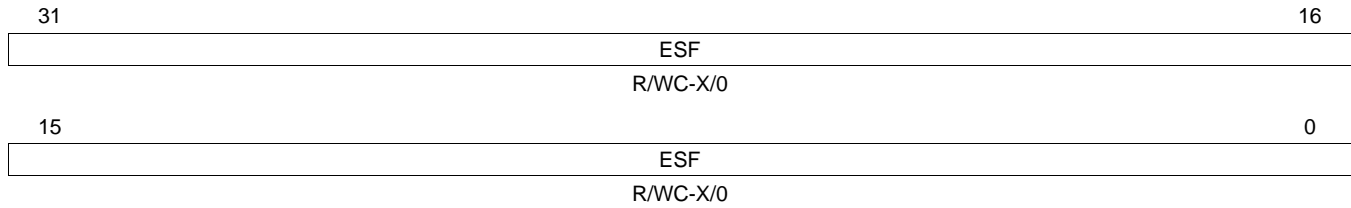
**Table 12-10. ESM Status Register 2 (ESMSR2) Field Descriptions**

Bit	Field	Value	Description
31-0	ESF	0	Error Status Flag. Provides status information on a pending error. <b>User and privileged mode:</b> Read: No error occurred; no interrupt is pending. <b>Privileged mode:</b> Write: Leaves the bit unchanged.
		1	<b>User and privileged mode:</b> Read: Error occurred; interrupt is pending. <b>Privileged mode:</b> Write: Clears the bit. ESMSSR2 is not impacted by this action. <b>Note:</b> In normal operation the flag gets cleared when reading the appropriate vector in the ESMIOFFHR offset register. Reading ESMIOFFHR will not clear the ESMSR1 and the shadow register ESMSSR2.

### 12.4.9 ESM Status Register 3 (ESMSR3)

The base address for this register is FFFF F520h. This register is dedicated for Group3.

**Figure 12-19. ESM Status Register 3 (ESMSR3)**



LEGEND: R = Read; W = Write; C = Clear; -n = value after reset/PORRST; X = Value unchanged

**Table 12-11. ESM Status Register 3 (ESMSR3) Field Descriptions**

Bit	Field	Value	Description
31-0	ESF	0	Error Status Flag. Provides status information on a pending error. <b>User and privileged mode:</b> Read: No error occurred. <b>Privileged mode:</b> Write: Leaves the bit unchanged.
		1	<b>User and privileged mode:</b> Read: Error occurred. <b>Privileged mode:</b> Write: Clears the bit.

### 12.4.10 ESM $\overline{\text{ERROR}}$ Pin Status Register (ESMEPSR)

The base address for this register is FFFF F524h.

**Figure 12-20. ESM  $\overline{\text{ERROR}}$  Pin Status Register (ESMEPSR)**

31	Reserved	16
R-0		
15	Reserved	1 0
R-0		EPSF
		R-X/1

LEGEND: R = Read only; -n = value after reset/ $\overline{\text{PORRST}}$ ; X = Value unchanged

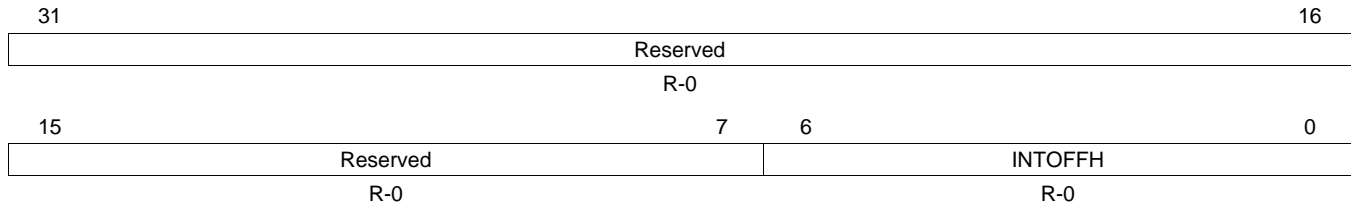
**Table 12-12. ESM  $\overline{\text{ERROR}}$  Pin Status Register (ESMEPSR) Field Descriptions**

Bit	Field	Value	Description
31-1	Reserved	0	Read returns 0. Writes have no effect.
0	EPSF	0	$\overline{\text{ERROR}}$ Pin Status Flag. Provides status information for the $\overline{\text{ERROR}}$ Pin. <b>User and privileged mode:</b> Read: $\overline{\text{ERROR}}$ Pin is low (active) if any error has occurred. Write: Writes have no effect.
		1	<b>User and privileged mode:</b> Read: $\overline{\text{ERROR}}$ Pin is high if no error has occurred. Write: Writes have no effect. <b>Note:</b> This flag will be set to 1 after PORRST. The value will be unchanged after nRST. The $\overline{\text{ERROR}}$ pin status remains un-changed during after nRST.

### 12.4.11 ESM Interrupt Offset High Register (ESMIOFFHR)

The base address for this register is FFFF F528h.

**Figure 12-21. ESM Interrupt Offset High Register (ESMIOFFHR)**



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

**Table 12-13. ESM Interrupt Offset High Register (ESMIOFFHR) Field Descriptions**

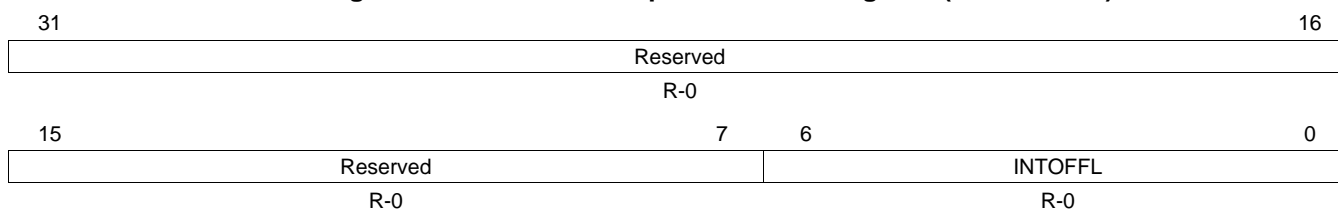
Bit	Field	Value	Description
31-7	Reserved	0	Read returns 0. Writes have no effect.
6-0	INTOFFH		<p>Offset High Level Interrupt. This vector gives the channel number of the highest pending interrupt request for the high level interrupt line. Interrupts of error Group2 have higher priority than interrupts of error Group1. Inside a group, channel 0 has highest priority and channel 31 has lowest priority.</p> <p><b>User and privileged mode (read):</b> Returns number of pending interrupt with the highest priority for the high level interrupt line.</p> <p style="margin-left: 20px;">0      No pending interrupt.</p> <p style="margin-left: 20px;">1h     Interrupt pending for channel 0, error Group1.</p> <p style="margin-left: 20px;">...</p> <p style="margin-left: 20px;">20h    Interrupt pending for channel 31, error Group1.</p> <p style="margin-left: 20px;">21h    Interrupt pending for channel 0, error Group2.</p> <p style="margin-left: 20px;">...</p> <p style="margin-left: 20px;">40h    Interrupt pending for channel 31, error Group2.</p> <p style="margin-left: 20px;">41h    Interrupt pending for channel 32, error Group1.</p> <p style="margin-left: 20px;">...</p> <p style="margin-left: 20px;">60h    Interrupt pending for channel 63, error Group1.</p> <p><b>Note:</b> Reading the interrupt vector will clear the corresponding flag in the ESMSR2 register; will <b>not</b> clear ESMSR1 and ESMSR2 and the offset register gets updated.</p> <p><b>User and privileged mode (write):</b> Writes have no effect.</p>



### 12.4.12 ESM Interrupt Offset Low Register (ESMIOFFLR)

The base address for this register is FFFF F52Ch.

**Figure 12-22. ESM Interrupt Offset Low Register (ESMIOFFLR)**



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

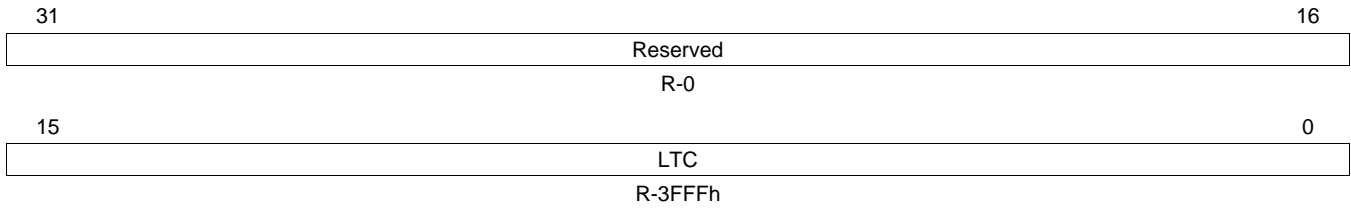
**Table 12-14. ESM Interrupt Offset Low Register (ESMIOFFLR) Field Descriptions**

Bit	Field	Value	Description
31-7	Reserved	0	Read returns 0. Writes have no effect.
6-0	INTOFFL	0	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.
			<b>User and privileged mode (read):</b>
			Returns number of pending interrupt with the highest priority for the low level interrupt line.
		0	No pending interrupt.
		1h	Interrupt pending for channel 0, error Group1.
		...	
		20h	Interrupt pending for channel 31, error Group1.
		21h	Interrupt pending for channel 32, error Group1.
		...	
		60h	Interrupt pending for channel 63, error Group1.
			<b>Note:</b> Reading the interrupt vector will <b>not</b> clear the corresponding flag in the ESMSR1 register. Group2 interrupts are fixed to the high level interrupt line only.
			<b>User and privileged mode (write):</b>
			Writes have no effect.

### 12.4.13 ESM Low-Time Counter Register (ESMLTCR)

The base address for this register is FFFF F530h.

**Figure 12-23. ESM Low-Time Counter Register (ESMLTCR)**



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

**Table 12-15. ESM Low-Time Counter Register (ESMLTCR) Field Descriptions**

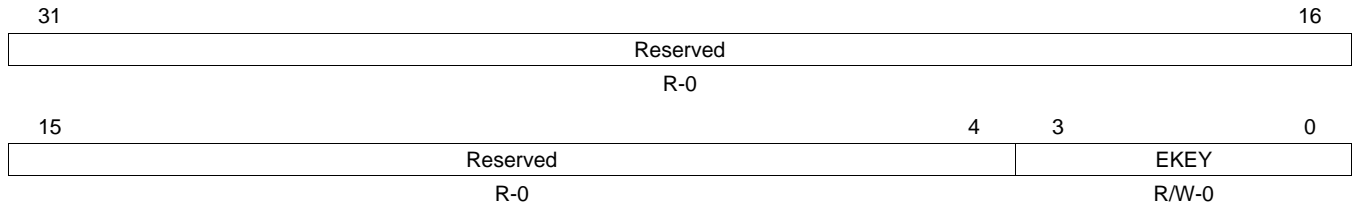
Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	LTC		ERROR Pin Low-Time Counter 16bit pre-loadable down-counter to control low-time of $\overline{\text{ERROR}}$ pin. The low-time counter is triggered by the peripheral clock (VCLK). <b>Note:</b> Low time counter is set to the default preload value of the ESMLTCPR in the following cases: <ol style="list-style-type: none"> <li>1. Reset (power on reset or warm reset)</li> <li>2. An error occurs</li> <li>3. User forces an error</li> </ol>



### 12.4.15 ESM Error Key Register (ESMEKR)

The base address for this register is FFFF F538h.

**Figure 12-25. ESM Error Key Register (ESMEKR)**



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

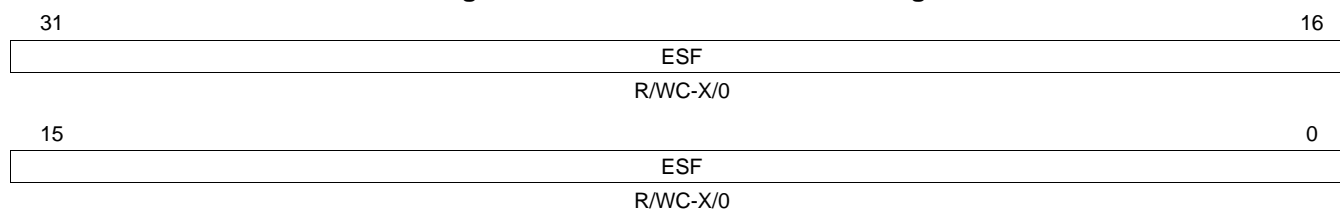
**Table 12-17. ESM Error Key Register (ESMEKR) Field Descriptions**

Bit	Field	Value	Description
31-4	Reserved	0	Read returns 0. Writes have no effect.
3-0	EKEY	0 Ah 5h  All other values	Error Key. The key to reset the $\overline{\text{ERROR}}$ pin or to force an error on the $\overline{\text{ERROR}}$ pin. <b>User and privileged mode (read):</b> Returns current value of the EKEY. <b>Privileged mode (write):</b> 0 Activates normal mode (recommended default mode). Ah Forces error on $\overline{\text{ERROR}}$ pin. 5h The $\overline{\text{ERROR}}$ pin set to high when the low time counter (LTC) has completed; then the EKEY bit will switch back to normal mode (EKEY = 0000) All other values Activates normal mode.

### 12.4.16 ESM Status Shadow Register 2 (ESMSSR2)

The base address for this register is FFFF F53Ch. This register is dedicated for Group2.

**Figure 12-26. ESM Status Shadow Register 2**



LEGEND: R = Read; W = Write; C = Clear; -n = value after reset/PORRST; X = Value unchanged

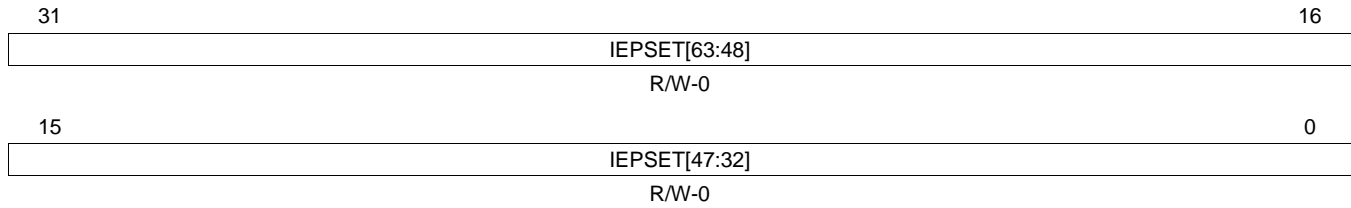
**Table 12-18. ESM Status Shadow Register 2 (ESMSSR2) Field Descriptions**

Bit	Field	Value	Description
31-0	ESF	0	Error Status Flag. Shadow register for status information on pending error. <b>User and privileged mode:</b> Read: No error occurred. <b>Privileged mode:</b> Write: Leaves the bit unchanged.
		1	<b>User and privileged mode:</b> Read: Error occurred. <b>Privileged mode:</b> Write: Clears the bit. ESMSSR2 is not impacted by this action. <b>Note:</b> Errors are stored until they are cleared by the software or at power-on reset (PORRST).

### 12.4.17 ESM Influence $\overline{\text{ERROR}}$ Pin Set/Status Register 4 (ESMIEPSR4)

The base address for this register is FFFF F540h. This register is dedicated for Group1.

**Figure 12-27. ESM Influence  $\overline{\text{ERROR}}$  Pin Set/Status Register 4 (ESMIEPSR4)**



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

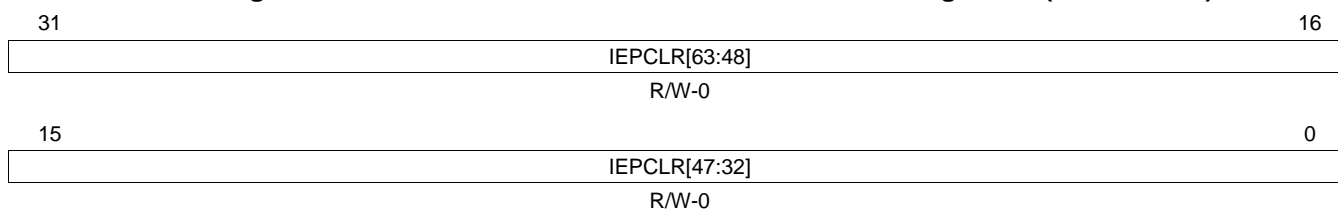
**Table 12-19. ESM Influence  $\overline{\text{ERROR}}$  Pin Set/Status Register 4 (ESMIEPSR4) Field Descriptions**

Bit	Field	Value	Description
63-32	IEPSET	0	Set Influence on $\overline{\text{ERROR}}$ Pin <b>User and privileged mode:</b> Read: Failure on channel x has no influence on $\overline{\text{ERROR}}$ pin. <b>Privileged mode:</b> Write: Leaves the bit and the corresponding clear bit in the ESMIEPCR4 register unchanged.
		1	<b>User and privileged mode:</b> Read: Failure on channel x has influence on $\overline{\text{ERROR}}$ pin. <b>Privileged mode:</b> Write: Enables failure influence on $\overline{\text{ERROR}}$ pin and sets the corresponding clear bit in the ESMIEPCR4 register.

### 12.4.18 ESM Influence $\overline{\text{ERROR}}$ Pin Clear/Status Register 4 (ESMIEPCR4)

The base address for this register is FFFF F544h. This register is dedicated for Group1.

**Figure 12-28. ESM Influence  $\overline{\text{ERROR}}$  Pin Clear/Status Register 4 (ESMIEPSR4)**



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

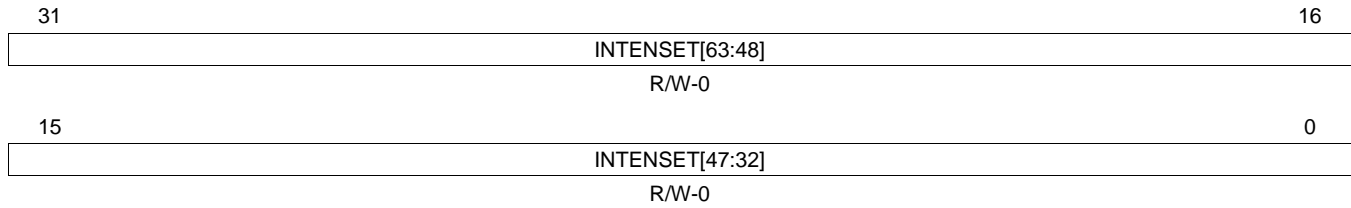
**Table 12-20. ESM Influence  $\overline{\text{ERROR}}$  Pin Clear/Status Register 4 (ESMIEPSR4) Field Descriptions**

Bit	Field	Value	Description
63-32	IEPCLR	0	Clear Influence on $\overline{\text{ERROR}}$ Pin <b>User and privileged mode:</b> Read: Failure on channel x has no influence on $\overline{\text{ERROR}}$ pin. <b>Privileged mode:</b> Write: Leaves the bit and the corresponding clear bit in the ESMIEPSR4 register unchanged.
		1	<b>User and privileged mode:</b> Read: Failure on channel x has influence on $\overline{\text{ERROR}}$ pin. <b>Privileged mode:</b> Write: Disables failure influence on $\overline{\text{ERROR}}$ pin and clears the corresponding clear bit in the ESMIEPSR4 register.

### 12.4.19 ESM Interrupt Enable Set/Status Register 4 (ESMIESR4)

The base address for this register is FFFF F548h. This register is dedicated for Group1.

**Figure 12-29. ESM Interrupt Enable Set/Status Register 4 (ESMIESR4)**



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

**Table 12-21. ESM Interrupt Enable Set/Status Register 4 (ESMIESR4) Field Descriptions**

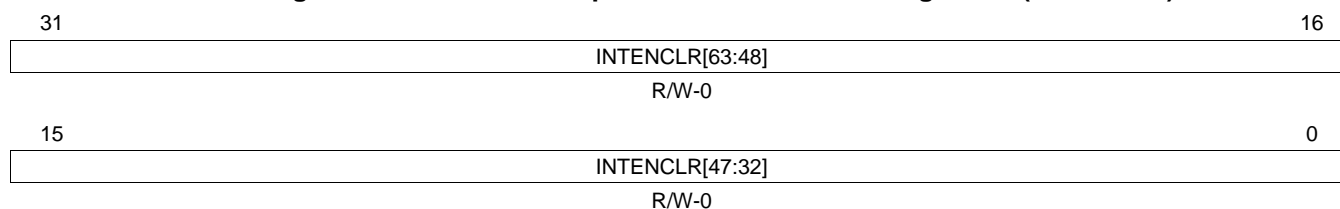
Bit	Field	Value	Description
63-32	INTENSET	0	Set Interrupt Enable. <b>User and privileged mode:</b> Read: Interrupt is disabled <b>Privileged mode:</b> Write: Leaves the bit and the corresponding clear bit in the ESMIECR4 register unchanged.
		1	<b>User and privileged mode:</b> Read: Interrupt is enabled. <b>Privileged mode:</b> Write: Enables interrupt and sets the corresponding clear bit in the ESMIECR4 register.



### 12.4.20 ESM Interrupt Enable Clear/Status Register 4 (ESMIECR4)

The base address for this register is FFFF F54Ch. This register is dedicated for Group1.

**Figure 12-30. ESM Interrupt Enable Clear/Status Register 4 (ESMIECR4)**



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

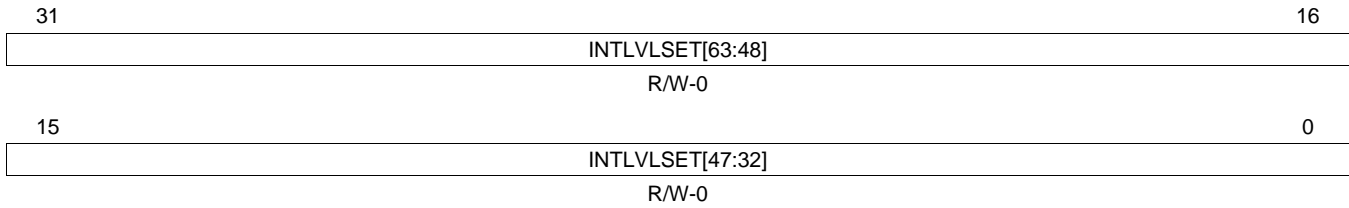
**Table 12-22. ESM Interrupt Enable Clear/Status Register 4 (ESMIECR4) Field Descriptions**

Bit	Field	Value	Description
63-32	INTENCLR	0	Clear Interrupt Enable. <b>User and privileged mode:</b> Read: Interrupt is disabled.
		1	<b>Privileged mode:</b> Write: Leaves the bit and the corresponding clear bit in the ESMIESR4 register unchanged. <b>User and privileged mode:</b> Read: Interrupt is enabled. <b>Privileged mode:</b> Write: Disables interrupt and clears the corresponding clear bit in the ESMIESR4 register.

### 12.4.21 ESM Interrupt Level Set/Status Register 4 (ESMILSR4)

The base address for this register is FFFF F550h. This register is dedicated for Group1.

**Figure 12-31. ESM Interrupt Level Set/Status Register 4 (ESMILSR4)**



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

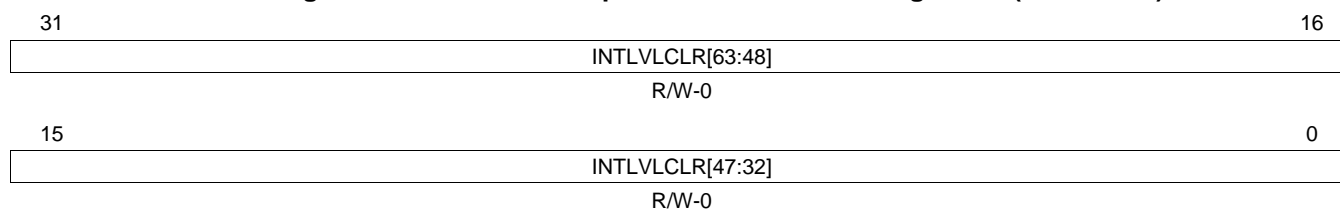
**Table 12-23. ESM Interrupt Level Set/Status Register 4 (ESMILSR4) Field Descriptions**

Bit	Field	Value	Description
63-32	INTLVLSET	0	Set Interrupt Level. <b>User and privileged mode:</b> Read: Read: Interrupt of channel x is mapped to low level interrupt line. <b>Privileged mode:</b> Write: Leaves the bit and the corresponding clear bit in the ESMILCR4 register unchanged.
		1	<b>User and privileged mode:</b> Read: Interrupt of channel x is mapped to high level interrupt line. <b>Privileged mode:</b> Write: Maps interrupt of channel x to high level interrupt line and sets the corresponding clear bit in the ESMILCR4 register.

### 12.4.22 ESM Interrupt Level Clear/Status Register 4 (ESMILCR4)

The base address for this register is FFFF F554h. This register is dedicated for Group1.

**Figure 12-32. ESM Interrupt Level Clear/Status Register 4 (ESMILCR4)**



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

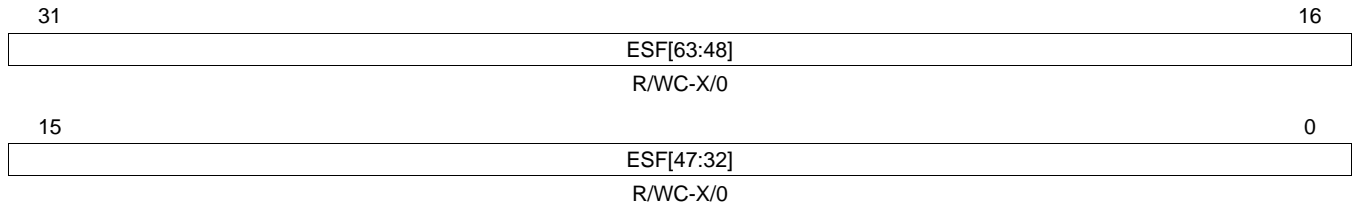
**Table 12-24. ESM Interrupt Level Clear/Status Register 4 (ESMILCR4) Field Descriptions**

Bit	Field	Value	Description
63-32	INTLVLCLR	0	Clear Interrupt Level. <b>User and privileged mode:</b> Read: Interrupt of channel x is mapped to low level interrupt line. <b>Privileged mode:</b> Write: Leaves the bit and the corresponding set bit in the ESMILSR4 register unchanged.
		1	<b>User and privileged mode:</b> Read: Interrupt of channel x is mapped to high level interrupt line. <b>Privileged mode:</b> Write: Maps interrupt of channel x to low level interrupt line and clears the corresponding set bit in the ESMILSR4 register.

### 12.4.23 ESM Status Register 4 (ESMSR4)

The base address for this register is FFFF F558h. This register is dedicated for Group1.

**Figure 12-33. ESM Status Register 4 (ESMSR4)**



LEGEND: R = Read; W = Write; C = Clear; -n = value after reset/PORRST; X = Value unchanged

**Table 12-25. ESM Status Register 4 (ESMSR4) Field Descriptions**

Bit	Field	Value	Description
63-32	ESF	0	<p>Error Status Flag. Provides status information on a pending error.</p> <p><b>User and privileged mode:</b>                      Read: No error occurred; no interrupt is pending.                      Write: Leaves the bit unchanged.</p> <p><b>Privileged mode:</b>                      Read: Error occurred; interrupt is pending.                      Write: Clears the bit.</p>
		1	<p><b>User and privileged mode:</b>                      Read: Error occurred; interrupt is pending.                      Write: Leaves the bit unchanged.</p> <p><b>Privileged mode:</b>                      Read: Error occurred; interrupt is pending.                      Write: Clears the bit.</p> <p><b>Note:</b> After <math>\overline{RST}</math>, if one of these flags are set and the corresponding interrupt are enabled, the interrupt service routine will be called.</p>



## ***Real-Time Interrupt (RTI) Module***

---

---

---

This chapter describes the functionality of the real-time interrupt (RTI) module. The RTI is designed as an operating system timer to support a real time operating system (RTOS).

<b>Topic</b>	<b>Page</b>
<b>13.1 Overview</b> .....	<b>460</b>
<b>13.2 Module Operation</b> .....	<b>461</b>
<b>13.3 Control Registers</b> .....	<b>471</b>

## 13.1 Overview

The real-time interrupt (RTI) module provides timer functionality for operating systems and for benchmarking code. The RTI module can incorporate several counters that define the timebases needed for scheduling in the operating system.

The timers also allow you to benchmark certain areas of code by reading the values of the counters at the beginning and the end of the desired code range and calculating the difference between the values.

### 13.1.1 Features

The RTI module has the following features:

- Two independent 64 bit counter blocks
- Four configurable compares for generating operating system ticks or DMA requests. Each event can be driven by either counter block 0 or counter block 1.
- 
- Fast enabling/disabling of events
- Two time stamp (capture) functions for system or peripheral interrupts, one for each counter block
- Digital windowed watchdog

### 13.1.2 Industry Standard Compliance Statement

This module is specifically designed to fulfill the requirements for OSEK (**O**ffene **S**ysteme und deren **S**chnittstellen für die **E**lektronik im **K**raftfahrzeug, or Open Systems and the Corresponding Interfaces for Automotive Electronics) as well as OSEK/time-compliant operating systems, but is not limited to it.

### 13.2 Module Operation

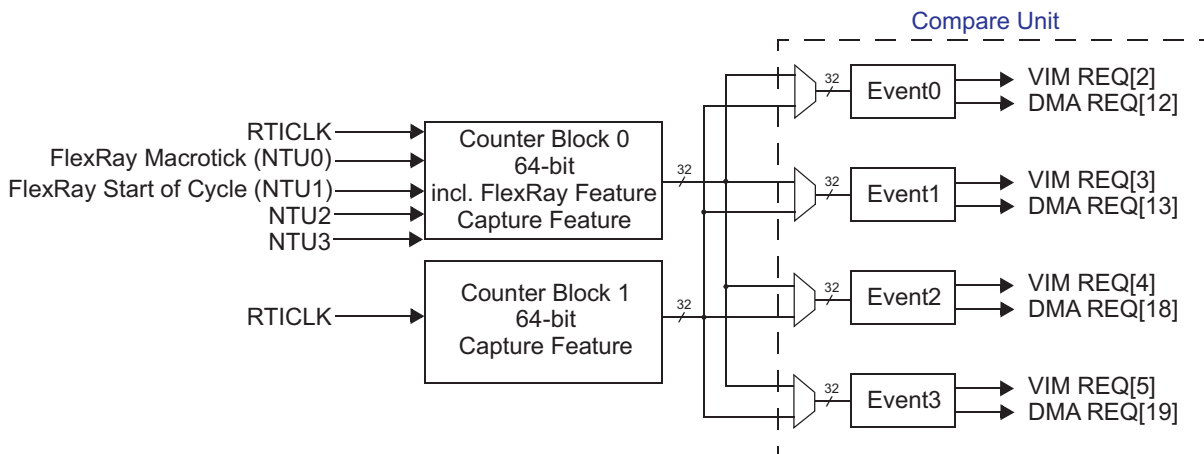
Figure 13-1 illustrates the high level block diagram of the RTI module.

The RTI module has two independent counter blocks for generating different timebases: counter block 0 and counter block 1.

A compare unit compares the counters with programmable values and generates four independent interrupt or DMA requests on compare matches. Each of the compare registers can be programmed to be compared to either counter block 0 or counter block 1.

The following sections describe the individual functions in more detail.

**Figure 13-1. RTI Block Diagram**



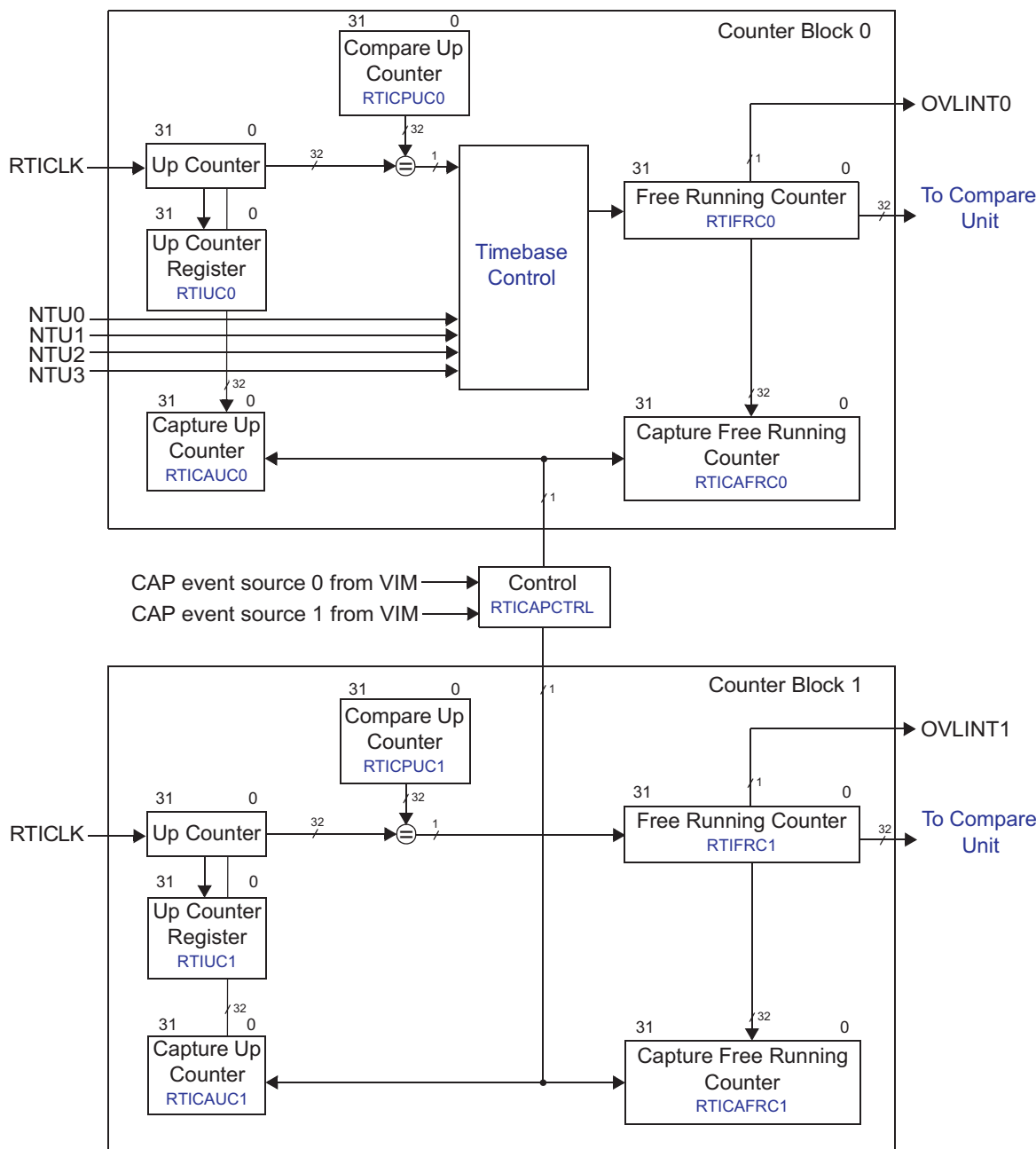
#### 13.2.1 Counter Operation

Each counter block consists of the following (see Figure 13-2):

- One 32-bit prescale counter (RTIUC0 or RTIUC1)
- One 32-bit free running counter (RTIFRC0 or RTIFRC1)



Figure 13-2. Counter Block Diagram



The RTIUC0/1 is driven by the RTICKL and counts up until the compare value in the compare up counter register (RTICPUC0 or RTICPUC1) is reached. When the compare matches, RTIFRC0/1 is incremented and RTIUC0/1 is reset to 0. If RTIFRC0/1 overflows, an interrupt is generated to the vectored interrupt manager (VIM). The overflow interrupt is not intended to generate the timebase for the operating system. See Section 13.2.2 for the timebase generation. The up counter together with the compare up counter value prescale the RTI clock. The resulting formula for the frequency of the free running counter (RTIFRC0/1) is:

$$f_{RTIFRCx} = \begin{cases} \frac{f_{RTICKL}}{RTICPUCx + 1} & \text{when } RTICPUCx \neq 0 \\ \frac{f_{RTICKL}}{2^{32}} & \text{when } RTICPUCx = 0 \end{cases} \quad (23)$$

The counter values can be determined by reading the respective counter registers or by generating a hardware event which captures the counter value into the respective capture register. Both functions are described in the following sections.

### 13.2.1.1 Counter and Capture Read Consistency

Portions of the device internal databus is 32-bits wide. If the application wants to read the 64-bit counters or the 64-bit capture values, a certain order of 32-bit read operations needs to be followed. This is to prevent one counter incrementing in between the two separate read operations to both counters.

#### Reading the Counters

The free running counter (RTIFRCx) must be read first. This priority will ensure that in the cycle when the CPU reads RTIFRCx, the up counter value is stored in its counter register (RTIUCx). The second read has to access the up counter register (RTIUCx), which then holds the value which corresponds to the number of RTICK cycles that have elapsed at the time reading the free running counter register (RTIFRCx).

---

**NOTE:** The up counters are implemented as shadow registers. Reading RTIUCx without having read RTIFRCx first will return always the same value. RTIUCx will only be updated when RTIFRCx is read.

---

#### Reading the Capture Values

The free running counter capture register (RTICAFRCx) must be read first. This priority will ensure that in the cycle when the CPU reads RTICAFRCx, the up counter value is stored in its counter register (RTICAUCx). The second read has to access the up counter register (RTICAUCx), which then holds the value captured at the time when reading the capture free running counter register (RTICAFRCx).

---

**NOTE:** The capture up counter registers are implemented as shadow registers. Reading RTICAUCx without having read RTICAFRCx first will return always the same value. RTICAUCx will only be updated when RTICAFRCx is read.

---

### 13.2.1.2 Capture Feature

Both counter blocks also provide a capture feature on external events. Two capture sources can trigger the capture event. The source triggering the block is configurable (RTICAPCTRL). The sources originate from the Vectored Interrupt Manager (VIM) and allow the generation of capture events when a peripheral module has generated an interrupt. Any of the peripheral interrupts can be selected as the capture event in the VIM.

When an event is detected, RTIUCx and RTIFRCx are stored in the capture up counter (RTICAUCx) and capture free running counter (RTICAFRCx) registers. The read order of the captured values must be the same as the read order of the actual counters (see [Section 13.2.1.1](#)).

### 13.2.2 Interrupt/DMA Requests

There are four compare registers (RTICOMP<sub>y</sub>) to generate interrupt requests to the VIM or DMA requests to the DMA controller. The interrupts can be used to generate different timebases for the operating system. Each of the compare registers can be configured to be compared to either RTIFRC0 or RTIFRC1. When the counter value matches the compare value, an interrupt is generated. To allow periodic interrupts a certain value can be added to the compare value in RTICOMP<sub>y</sub> automatically. This value is stored in the update compare register (RTIUDCP<sub>y</sub>) and will be added after a compare matched. The period of the generated interrupt/DMA request can be calculated with:

$$t_{\text{COMP}x} = t_{\text{RTICLK}} \times (\text{RTICPUCy} + 1) \times \text{RTIUDCPy}$$

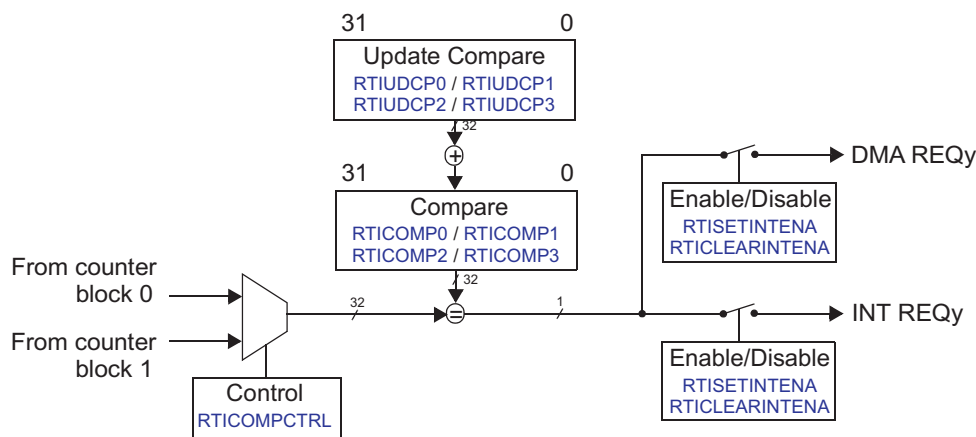
if  $\text{RTICPUCy} = 0$ ,

$$t_{\text{COMP}x} = t_{\text{RTICLK}} \times 2^{32} \times \text{RTIUDCPy}$$

if  $\text{RTIUDCPy} = 0$ ,

$$t_{\text{COMP}x} = t_{\text{RTICLK}} \times (\text{RTICPUCy} + 1) \times 2^{32} \quad (24)$$

**Figure 13-3. Compare Unit Block Diagram (shows only 1 of 4 blocks for simplification)**



Another interrupt that can be generated is the overflow interrupt (OVLINT<sub>x</sub>) in case the RTIFRC<sub>x</sub> counter overflows.

The interrupts/DMA requests can be enabled in the RTISETINTENA register and disabled in the RTICLEARINTENA register. The RTIINTFLAG register shows the pending interrupts.

### 13.2.3 RTI Clocking

The counter blocks are clocked with RTICLK (for definition see "Clock Domains").

A clock supervision for the NTU<sub>x</sub> clocking scheme is implemented to avoid missing operating system ticks.

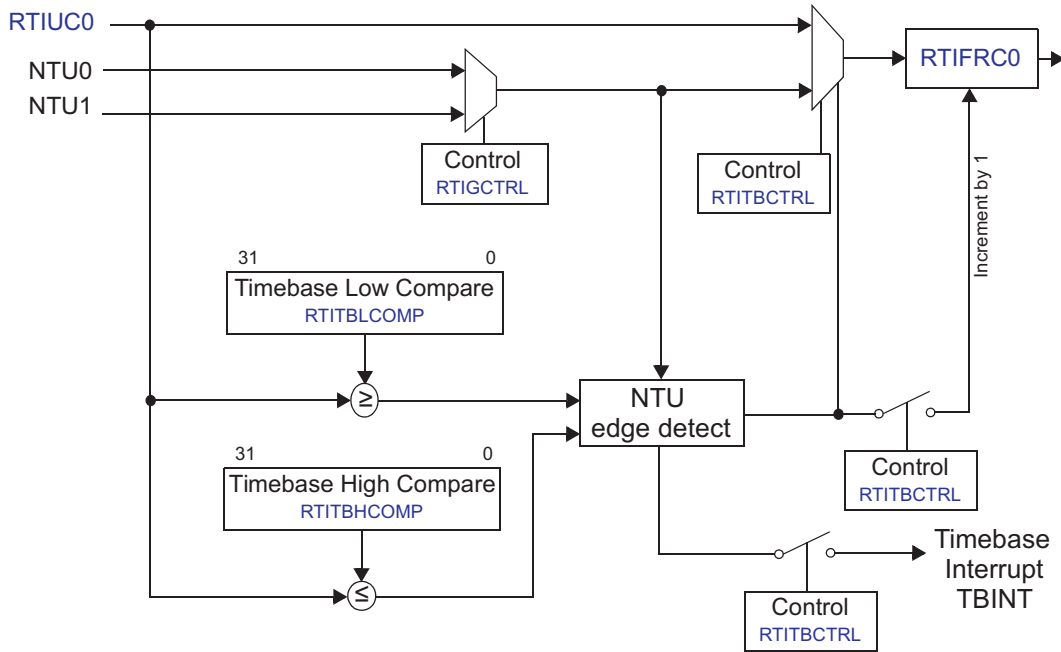
### 13.2.4 Synchronizing Timer Events to Network Time (NTU)

For applications which are participating on a time-triggered communication bus, it is often beneficial to synchronize the application or operating system to the network time. The RTI provides a feature to increment Free Running Counter 0 (RTIFRC0) by a periodic clock provided by the communication module. In this case two different clocks can be chosen.

The application has control over which clock (RTICLK, NTU0, NTU1) should be used for clocking RTIFRC0. If NTU<sub>x</sub> is used, a clock supervision circuit allows to monitor this clock and provides a fallback solution, should the clock be non-functional (missing). A too fast running NTU<sub>x</sub> cannot be detected.

RTIUC0 is utilized to monitor the NTU<sub>x</sub> signal. A detection window can be programmed in which a valid NTU clock pulse needs to occur. If no pulse is detected, the RTI automatically switches back to clock the Free Running Counter 0 with RTIUC0. In order to avoid a big jitter in the operating ticks, in case a switch back to RTIUC0 happens, RTICPUC0 should be set to a value so the clock frequency RTIUC0 outputs is approximately the same as the NTU<sub>x</sub> frequency.

Figure 13-4. Timebase Control



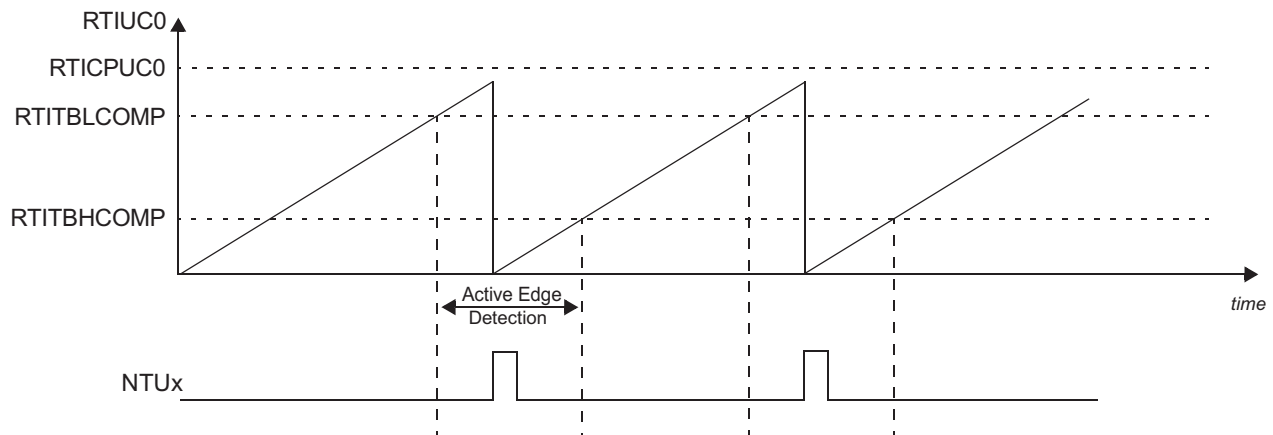
### 13.2.4.1 Detecting Clock Edges

To detect clock edges on the NTUx signal, the timebase low compare has to be set lower or equal than the value stored in the RTICPUC0 register and the timebase high compare has to be set higher than 0 and lower than the timebase low compare value. This effectively opens a window in which an edge of the NTUx signal is expected (see Figure 13-5). Outside this window, no edges will be detected. If no edge will occur inside the detection window, the multiplexer is switched to internal timebase. The application can select to generate a timebase interrupt (TBINT) and if the INC bit is set, also will automatically increment RTIFRC0 by one to compensate for the missed clock cycle of NTUx. If an edge occurs inside the window, RTIUC0 will be reset to synchronize the two timebases.

In order to make the edge detection work properly, the value in RTICPUC0 needs to be adapted so that RTIUC0 has a similar period as NTUx.

**NOTE:** To ensure the NTUx signal is properly detected, the NTUx period must be at least twice as long as the RTICLK period.

Figure 13-5. Clock Detection Scheme



### 13.2.4.2 Switching from Internal Source to External Source

If the application switches from an internal source to an external source, the two signals must be synchronized (see Figure 13-6). The synchronization will occur when the TBEXT bit is set. RTIUC0 will be reset and the edge detection circuit will be active for one (RTICPUC0 + RTITBHCMP) period or until an edge is detected. If there is no pulse during this period, the source will be reset from an external clock source to an internal clock source. If an edge is detected, the windowed edge detection behavior will take place. Setting the TBEXT bit will not increment free running counter 0.

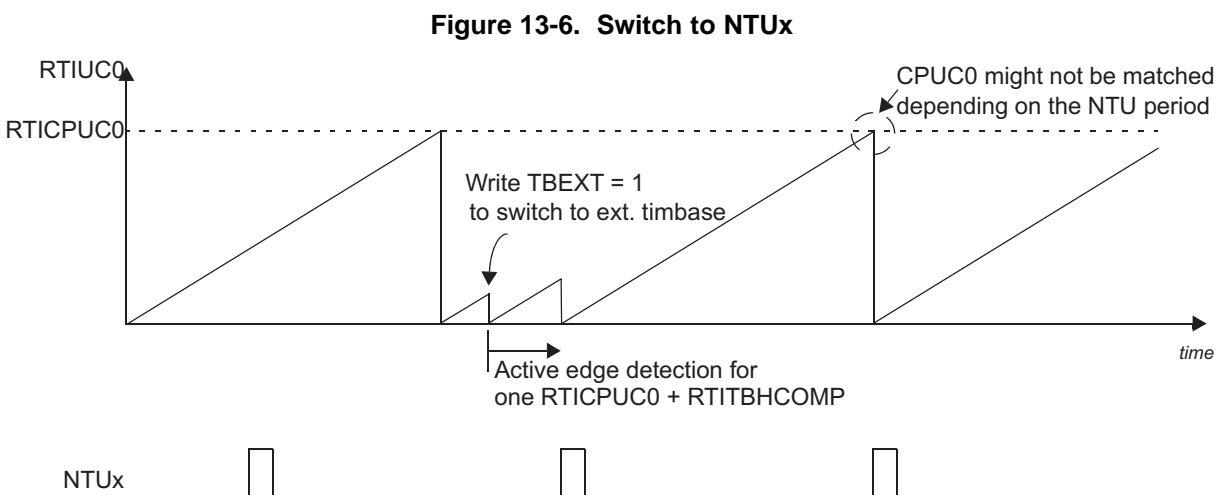
**NOTE:** If an external timebase is used, then the software must ensure that timebase low compare and timebase high compare are programmed to a valid state before switching TBEXT to an external source. This state is necessary to allow the timebase control circuit to operate correctly. The following condition must be met:

- $RTITBHCMP < RTITBLCOMP \leq RTICPUC0$

RTITBHCMP must be lower than RTICPUC0 because RTIUC0 will be reset if RTICPUC0 is reached. RTITBHCMP will represent the number of RTICLK cycles from RTICPUC0 until the circuit switches to the internal timebase when no NTU edge is detected.

If an external timebase is used, RTIGCTRL[0] must be set to 1 (enable RTIUC0) to ensure that the timebase control circuit does not wait indefinitely for an incoming signal.

Figure 13-6 shows a timing example for the synchronization phase when the TBEXT bit is set.

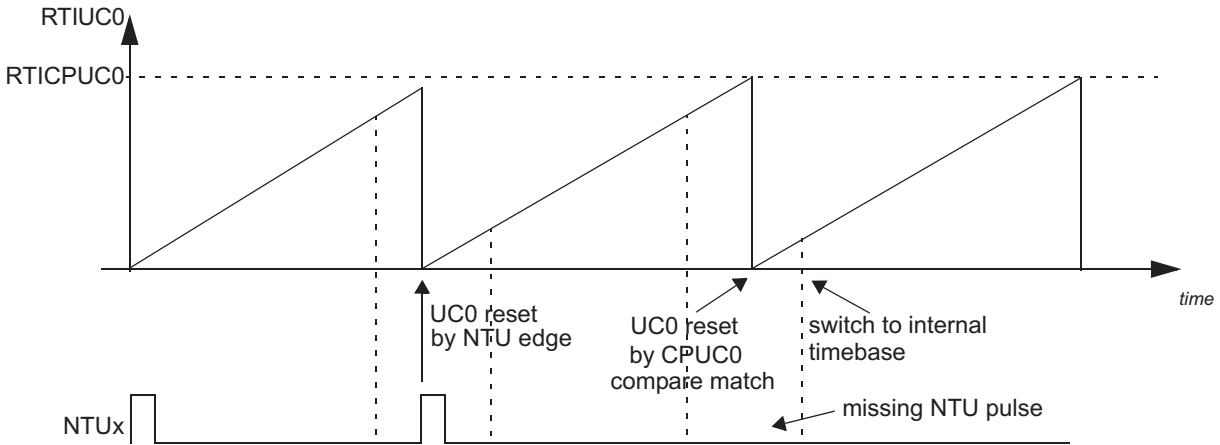


### 13.2.4.3 Switching from External Source to Internal Source

When the edge detection is active (TBEXT = 1) and no clock edge of NTUx is detected inside the programmed detection window, the RTI will automatically switch the timebase to RTIUC0. Figure 13-7 shows a timing example for a missing NTU signal. In the case where the INC bit is set, RTIFRC0 will automatically be incremented by one to compensate for the missed NTU pulse.

Setting TBEXT = 0 will also switch the clock source for RTIFRC0 to RTIUC0.

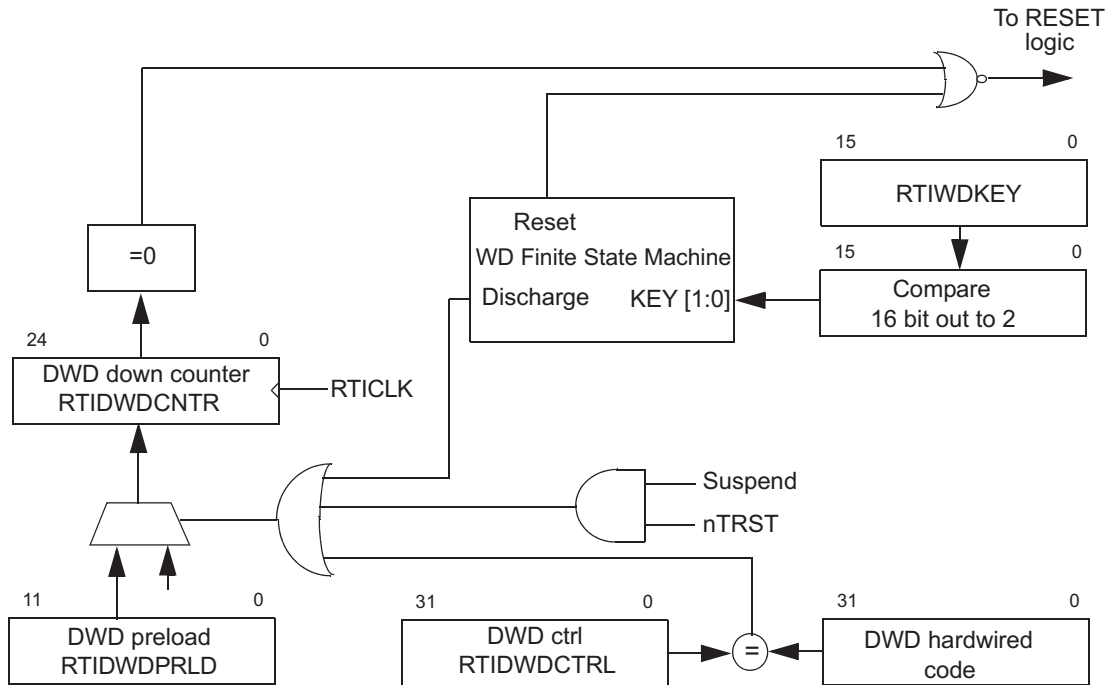
Figure 13-7. Missing NTUx Signal Example



### 13.2.5 Digital Watchdog (DWD)

The digital watchdog (DWD) is an optional safety diagnostic which can detect a runaway CPU and generate either a reset or NMI (non-maskable interrupt) response. It generates resets or NMIs after a programmable period, or if no correct key sequence was written to the RTIWDKEY register. Figure 13-8 illustrates the DWD.

Figure 13-8. Digital Watchdog



#### 13.2.5.1 Digital Watchdog (DWD)

The DWD is disabled by default. If it should be used, it must be enabled by writing a 32-bit value, which is the inverted value of the hardwired code in the module, to the RTIDWDCTRL register.

**NOTE:**

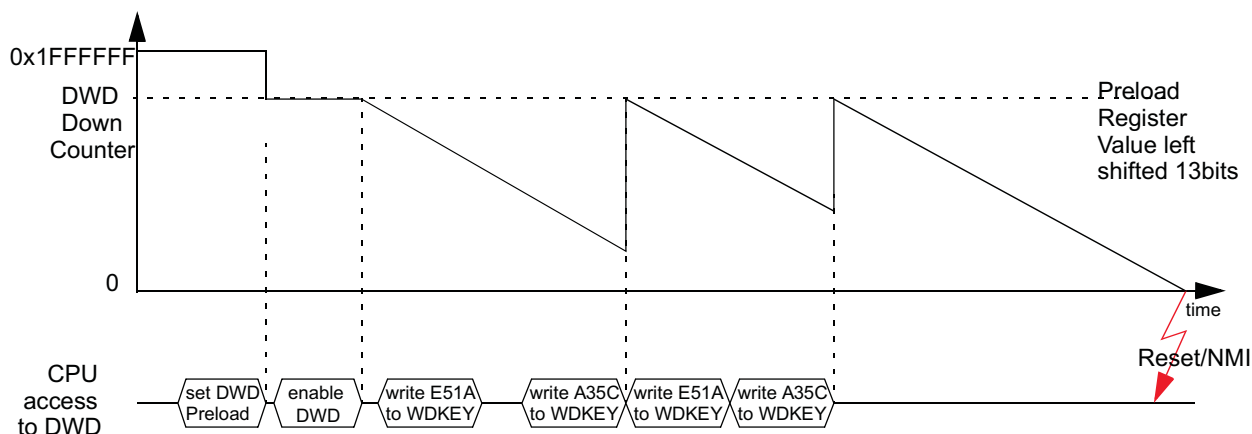
Once the DWD is enabled, it cannot be disabled except by system reset or power on reset.

If the correct key sequence is written to the RTIWDKEY register (0xE51A followed by 0xA35C), the 25-bit DWD down counter is reloaded with the left justified 12-bit preload value stored in RTIDWDPRLD. If an incorrect value is written, a watchdog reset or NMI will occur immediately. A reset or NMI will also be generated when the DWD down counter is decremented to 0.

While the device is in suspend mode (halting debug mode), the DWD down counter keeps the value it had when entering suspend mode.

The DWD down counter will be decremented with the RTICLK frequency.

**Figure 13-9. DWD Operation**



The expiration time of the DWD down counter can be determined with the following equation:

$$t_{exp} = (DWDPRLD + 1) \times 2^{13} / RTICLK$$

where

$$DWDPRLD = 0 \dots 4095$$

**NOTE:** Care should be taken to ensure that the CPU write to the watchdog register is made allowing time for the write to propagate to the RTI.

### 13.2.5.2 Digital Windowed Watchdog (DWWD)

In addition to the time-out boundary configurable via the digital watchdog discussed in [Section 13.2.5.1](#), for enhanced safety metrics it is desirable to check for a watchdog "pet" within a time window rather than using a single time threshold. This is enabled by the digital windowed watchdog (DWWD) feature.

- Functional Behavior

The DWWD opens a configurable time window in which the watchdog must be serviced. Any attempt to service the watchdog outside this time window, or a failure to service the watchdog in this time window, will cause the watchdog to generate either a reset or a NMI to the CPU. This is controlled by configuring the RTIWWDRXNCTRL register. As with the DWD, the DWWD is disabled after power on reset. When the DWWD is configured to generate a non-maskable interrupt on a window violation, the watchdog counter continues to count down. The NMI handler needs to clear the watchdog violation status flag(s) and then service the watchdog by writing the correct sequence in the watchdog key register. This service will cause the watchdog counter to get reloaded from the preload value and start counting down. If the NMI handler does not service the watchdog in time, it could count down all the way to zero and wrap around. If the NMI Handler does not service the watchdog in time, the NMI gets generated continuously, each time the counter counts to '0'.

The DWWD uses the Digital Watchdog (DWD) preload register (RTIDWDPRLD) setting to define the end-time of the window. The start-time of the window is defined by a window size configuration register (RTIWWDSIZECTRL).

The default window size is set to 100%, which corresponds to the DWD functionality of a time-out-only watchdog. The window size can be selected (through register RTIWWDSIZECTRL) from among 100%, 50%, 25%, 12.5%, 6.25% and 3.125% as shown in Figure 13-10. The window with the respective size will be opened before the end of the DWD expiration. The user has to serve the watchdog in the window. Otherwise, a reset or NMI will generate. Figure 13-11 shows an DWWD operation example (25% window).

- Configuration of DWWD

The DWWD preload value (same as DWD preload) can only be configured when the DWWD counter is disabled. The window size and watchdog reaction to a violation can be configured even after the watchdog has been enabled. Any changes to the window size and watchdog reaction configurations will only take effect after the next servicing of the DWWD. This feature can be utilized to dynamically set windows of different sizes based on task execution time, adding a program sequence element to the diagnostic which can improve fault coverage.

Figure 13-10. Digital Windowed Watchdog Timing Example

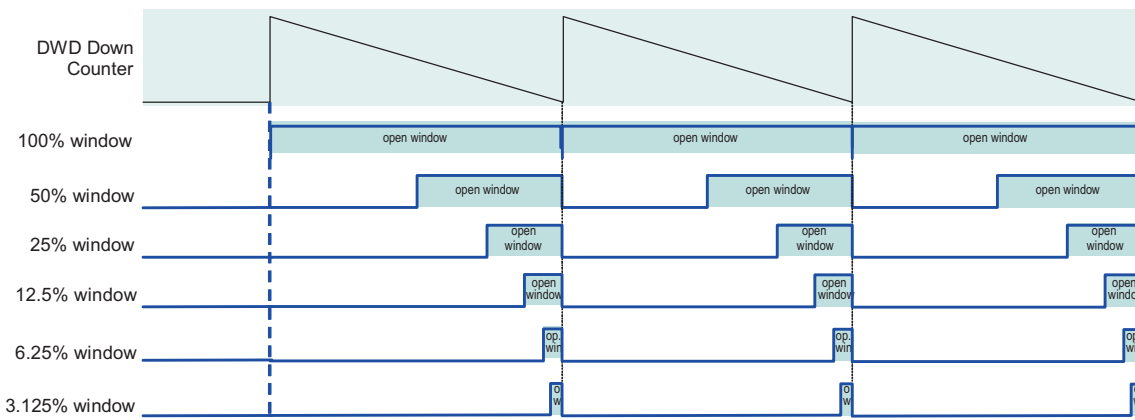
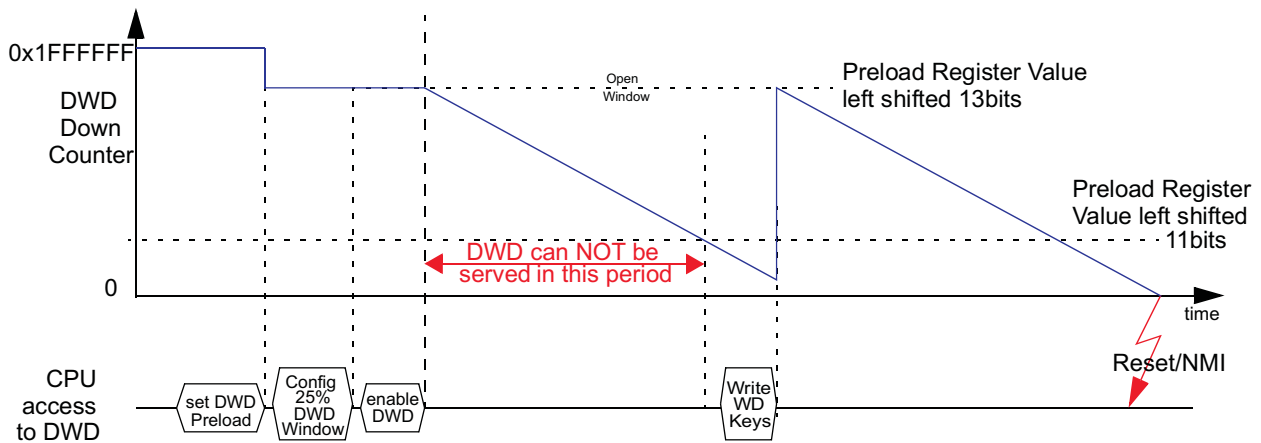


Figure 13-11. Digital Windowed Watchdog Operation Example (25% Window)



### 13.2.6 Low Power Modes

Low power modes allow the trade off of the current used during low power versus functionality and fast wakeup response. All low power modes have the following characteristics:

- CPU and system clocks are disabled.
- Flash banks and pump are in sleep mode.
- All peripheral modules are in low power modes and the clocks are disabled (exceptions to this may occur and would be documented in the specific device data sheet).

Flexibility in enabling and disabling clocks allows for many different low-power modes.



The operation of the RTI Module is guaranteed in Run, Doze and Snooze modes. In Sleep mode all clocks will be switched off and the RTI will not work.

In Doze and Snooze modes the RTI is active and is able to wake up the device with compare, timebase and overflow interrupts. The compare interrupts can be used to periodically wake up the device. The overflow interrupt can be used to notify the operating system that a counter overflow has occurred. Capturing events generated by the Vectored Interrupt Module (VIM) is also possible since in both these low power modes, the peripheral modules are able to generate interrupts that can trigger capture events. Capturing events while in Sleep mode is not supported as the clock to the RTI is not active.

When the device is put into low power mode, the peripheral which is generating the external clock NTU is no longer active, and the timebase control circuitry has to switch to an internal clocking scheme when it detects a missing clock on NTU. The timebase interrupt will wake up the device and the application software has to adapt the periodic interrupt generation to the internal clock source.

DMA transfers will be disabled, and DMA requests will not be generated after device wakeup since the DMA controller will be powered down.

---

**NOTE: RTICK in Doze Mode**

In the special case of Doze Mode with PLL off, RTICK might have a different period than with PLL enabled since RTICK will be derived from the oscillator output. It has to be ensured that the VCLK to RTICK ratio is at least 3:1.

---

### 13.2.7 Halting Debug Mode Behaviour

Once the system enters halting debug mode, the behaviour of the RTI depends on the COS (continue on suspend) bit. If the bit is cleared and halting debug mode is active, all counters will stop operation. If the bit is set to one, all counters will be clocked normally and the RTI will work like in normal mode. However if the external timebase (NTU) is used and the system is in halting debug mode, the timebase control circuit will switch to internal timebase once it detects the missing NTU signal of the suspended communication controller. This will be signaled with an TBINT interrupt, so that software can resynchronize after the device exits halting debug mode.

### 13.3 Control Registers

[Table 13-1](#) provides a summary of the registers. The registers support 8-bit, 16-bit, and 32-bit writes. The offset is relative to the associated peripheral select. See the following sections for detailed descriptions of the registers. The base address for the control registers is FFFF FC00h. The address locations not listed, are reserved..

**Table 13-1. RTI Registers**

Offset	Acronym	Register Description	Section
00	RTIGCTRL	RTI Global Control Register	<a href="#">Section 13.3.1</a>
04h	RTITBCTRL	RTI Timebase Control Register	<a href="#">Section 13.3.2</a>
08h	RTICAPCTRL	RTI Capture Control Register	<a href="#">Section 13.3.3</a>
0Ch	RTICOMPCTRL	RTI Compare Control Register	<a href="#">Section 13.3.4</a>
10h	RTIFRC0	RTI Free Running Counter 0 Register	<a href="#">Section 13.3.5</a>
14h	RTIUC0	RTI Up Counter 0 Register	<a href="#">Section 13.3.6</a>
18h	RTICPUC0	RTI Compare Up Counter 0 Register	<a href="#">Section 13.3.7</a>
20h	RTICAFRC0	RTI Capture Free Running Counter 0 Register	<a href="#">Section 13.3.8</a>
24h	RTICAUC0	RTI Capture Up Counter 0 Register	<a href="#">Section 13.3.9</a>
30h	RTIFRC1	RTI Free Running Counter 1 Register	<a href="#">Section 13.3.10</a>
34h	RTIUC1	RTI Up Counter 1 Register	<a href="#">Section 13.3.11</a>
38h	RTICPUC1	RTI Compare Up Counter 1 Register	<a href="#">Section 13.3.12</a>
40h	RTICAFRC1	RTI Capture Free Running Counter 1 Register	<a href="#">Section 13.3.13</a>
44h	RTICAUC1	RTI Capture Up Counter 1 Register	<a href="#">Section 13.3.14</a>
50h	RTICOMP0	RTI Compare 0 Register	<a href="#">Section 13.3.15</a>
54h	RTIUDCP0	RTI Update Compare 0 Register	<a href="#">Section 13.3.16</a>
58h	RTICOMP1	RTI Compare 1 Register	<a href="#">Section 13.3.17</a>
5Ch	RTIUDCP1	RTI Update Compare 1 Register	<a href="#">Section 13.3.18</a>
60h	RTICOMP2	RTI Compare 2 Register	<a href="#">Section 13.3.19</a>
64h	RTIUDCP2	RTI Update Compare 2 Register	<a href="#">Section 13.3.20</a>
68h	RTICOMP3	RTI Compare 3 Register	<a href="#">Section 13.3.21</a>
6Ch	RTIUDCP3	RTI Update Compare 3 Register	<a href="#">Section 13.3.22</a>
70h	RTITBLCOMP	RTI Timebase Low Compare Register	<a href="#">Section 13.3.23</a>
74h	RTITBHCOMP	RTI Timebase High Compare Register	<a href="#">Section 13.3.24</a>
80h	RTISETINTENA	RTI Set Interrupt Enable Register	<a href="#">Section 13.3.25</a>
84h	RTICLEARINTENA	RTI Clear Interrupt Enable Register	<a href="#">Section 13.3.26</a>
88h	RTIINTFLAG	RTI Interrupt Flag Register	<a href="#">Section 13.3.27</a>
90h	RTIDWDCTRL	Digital Watchdog Control Register	<a href="#">Section 13.3.28</a>
94h	RTIDWDPRLD	Digital Watchdog Preload Register	<a href="#">Section 13.3.29</a>
98h	RTIWDSTATUS	Watchdog Status Register	<a href="#">Section 13.3.30</a>
9Ch	RTIWDKEY	RTI Watchdog Key Register	<a href="#">Section 13.3.31</a>
A0h	RTIDWDCNTR	RTI Digital Watchdog Down Counter Register	<a href="#">Section 13.3.32</a>
A4h	RTIWWDRXNCTRL	Digital Windowed Watchdog Reaction Control Register	<a href="#">Section 13.3.33</a>
A8h	RTIWWDSIZECTRL	Digital Windowed Watchdog Window Size Control Register	<a href="#">Section 13.3.34</a>
ACh	RTIINTCLRENABLE	Do NOT use. Write to this register may clear the pending RTI interrupt.	
B0h	RTICOMP0CLR	Do NOT use. Write to this register may clear the pending RTI interrupt.	
B4h	RTICOMP1CLR	Do NOT use. Write to this register may clear the pending RTI interrupt.	
B8h	RTICOMP2CLR	Do NOT use. Write to this register may clear the pending RTI interrupt.	
BCh	RTICOMP3CLR	Do NOT use. Write to this register may clear the pending RTI interrupt.	



### 13.3.1 RTI Global Control Register (RTIGCTRL)

The global control register starts/stops the counters and selects the signal compared with the timebase control circuit. This register is shown in [Figure 13-12](#) and described in [Table 13-2](#).

**Figure 13-12. RTI Global Control Register (RTIGCTRL) [offset = 00]**

31	Reserved	20	19	18	17	16
		NTUSEL				
	R-0				R/W-0	
15	14			2	1	0
COS	Reserved			CNT1EN	CNT0EN	
R/W-0	R-0			R/W-0	R/W-0	

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

**Table 13-2. RTI Global Control Register (RTIGCTRL) Field Descriptions**

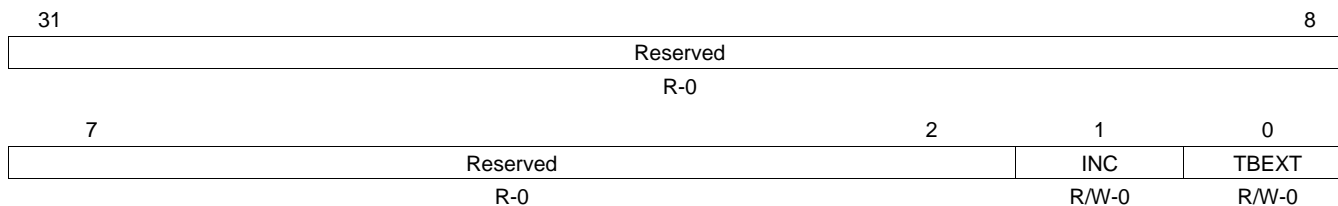
Bit	Field	Value	Description
31-20	Reserved	0	Read returns 0. Writes have no effect.
19-16	NTUSEL	0h 5h Ah Fh Others	Select NTU signal. These bits determine which NTU input signal is used as external timebase NTU0 NTU1 NTU2 NTU3 tied to '0'
15	COS	0 1	Continue on suspend. This bit determines if both counters are stopped when the device goes into halting debug mode or if they continue counting. Counters are stopped while in halting debug mode. Counters are running while in halting debug mode.
14-2	Reserved	0	Read returns 0. Writes have no effect.
1	CNT1EN	0 1	Counter 1 enable. This bit starts and stops counter block 1 (RTIUC1 and RTIFRC1). Counter block 1 is stopped. Counter block 1 is running.
0	CNT0EN	0 1	Counter 0 enable. This bit starts and stops counter block 0 (RTIUC0 and RTIFRC0). Counter block 0 is stopped. Counter block 0 is running.

**NOTE:** If the application uses the timebase circuit for synchronization between the communications controller and the operating system and the device enters halting debug mode, the synchronization may be lost depending on the COS setting in the RTI module and the halting debug mode behavior of the communications controller.

### 13.3.2 RTI Timebase Control Register (RTITBCTRL)

The timebase control register selects if the free running counter 0 is incremented by RTICKL or NTU. This register is shown in [Figure 13-13](#) and described in [Table 13-3](#).

**Figure 13-13. RTI Timebase Control Register (RTITBCTRL) [offset = 04h]**



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

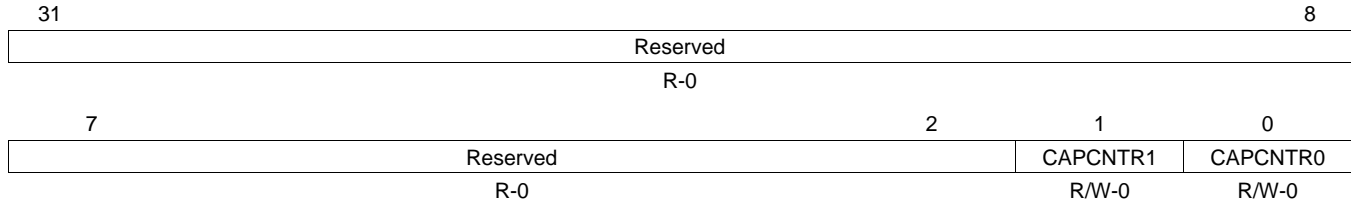
**Table 13-3. RTI Timebase Control Register (RTITBCTRL) Field Descriptions**

Bit	Field	Value	Description
31-2	Reserved	0	Read returns 0. Writes have no effect.
1	INC	0	Increment free running counter 0. This bit determines whether the free running counter 0 (RTIFRC0) is automatically incremented if a failing clock on the NTU signal is detected.
		1	RTIFRC0 will not be incremented on a failing external clock.
		1	RTIFRC0 will be incremented on a failing external clock.
0	TBEXT	0	Timebase external. This bit selects whether the free running counter 0 (RTIFRC0) is clocked by the internal up counter 0 (RTIUC0) or from the external signal NTU. Setting the TBEXT bit from 0 to 1 will not increment RTIFRC0, since RTIUC0 is reset.
		1	When the timebase supervisor circuit detects a missing clock edge, then the TBEXT bit is reset. Only the software can select whether the external signal should be used.
		0	RTIUC0 clocks RTIFRC0.
		1	NTU clocks RTIFRC0.

### 13.3.3 RTI Capture Control Register (RTICAPCTRL)

The capture control register controls the capture source for the counters. This register is shown in [Figure 13-14](#) and described in [Table 13-4](#).

**Figure 13-14. RTI Capture Control Register (RTICAPCTRL) [offset = 08h]**



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

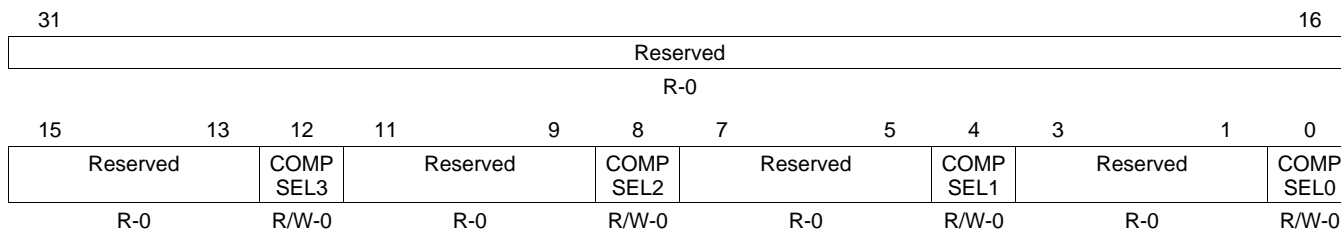
**Table 13-4. RTI Capture Control Register (RTICAPCTRL) Field Descriptions**

Bit	Field	Value	Description
31-2	Reserved	0	Read returns 0. Writes have no effect.
1	CAPCNTR1	0	Capture counter 1. This bit determines which external interrupt source triggers a capture event of RTIUC1 and RTIFRC1. 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	0	Capture counter 0. This bit determines which external interrupt source triggers a capture event of RTIUC0 and RTIFRC0. Capture of RTIUC0/ RTIFRC0 is triggered by capture event source 0.
		1	Capture of RTIUC0/ RTIFRC0 is triggered by capture event source 1.

### 13.3.4 RTI Compare Control Register (RTICOMPCTRL)

The compare control register controls the source for the compare registers. This register is shown in [Figure 13-15](#) and described in [Figure 13-15](#).

**Figure 13-15. RTI Compare Control Register (RTICOMPCTRL) [offset = 0Ch]**



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

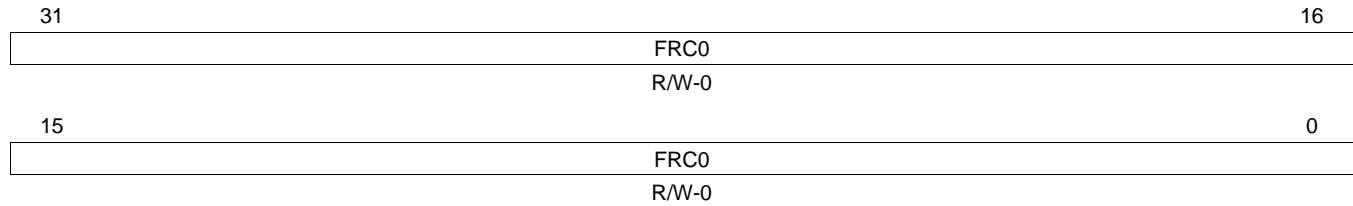
**Table 13-5. RTI Compare Control Register (RTICOMPCTRL) Field Descriptions**

Bit	Field	Value	Description
31-13	Reserved	0	Read returns 0. Writes have no effect.
12	COMPSEL3	0	Compare select 3. This bit determines the counter with which the compare value held in compare register 3 (RTICOMP3) is compared. Value will be compared with RTIFRC0.
		1	Value will be compared with RTIFRC1.
11-9	Reserved	0	Read returns 0. Writes have no effect.
8	COMPSEL2	0	Compare select 2. This bit determines the counter with which the compare value held in compare register 2 (RTICOMP2) is compared. Value will be compared with RTIFRC0.
		1	Value will be compared with RTIFRC1.
7-5	Reserved	0	Read returns 0. Writes have no effect.
4	COMPSEL1	0	Compare select 1. This bit determines the counter with which the compare value held in compare register 1 (RTICOMP1) is compared. Value will be compared with RTIFRC0.
		1	Value will be compared with RTIFRC1.
3-1	Reserved	0	Read returns 0. Writes have no effect.
0	COMPSEL0	0	Compare select 0. This bit determines the counter with which the compare value held in compare register 0 (RTICOMP0) is compared. Value will be compared with RTIFRC0.
		1	Value will be compared with RTIFRC1.

### 13.3.5 RTI Free Running Counter 0 Register (RTIFRC0)

The free running counter 0 register holds the current value of free running counter 0. This register is shown in [Figure 13-16](#) and described in [Table 13-6](#).

**Figure 13-16. RTI Free Running Counter 0 Register (RTIFRC0) [offset = 10h]**



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

**Table 13-6. RTI Free Running Counter 0 Register (RTIFRC0) Field Descriptions**

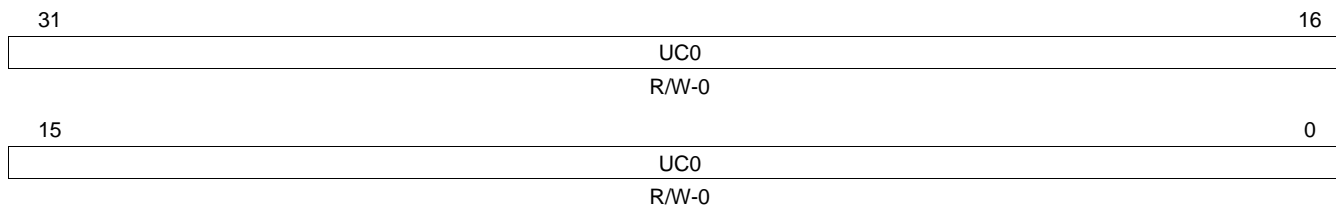
Bit	Field	Value	Description
31-0	FRC0	0-FFFF FFFFh	<p>Free running counter 0. This registers holds the current value of the free running counter 0. A read of this counter returns the current value of the counter.</p> <p>The counter can be preset by writing (in privileged mode only) to this register. The counter increments then from this written value upwards.</p> <p><b>Note: If counters must be preset, they must be disabled in the RTIGCTRL register to ensure consistency between RTIUC0 and RTIFRC0.</b></p>



### 13.3.6 RTI Up Counter 0 Register (RTIUC0)

The up counter 0 register holds the current value of prescale counter. This register is shown in [Figure 13-17](#) and described in [Table 13-7](#).

**Figure 13-17. RTI Up Counter 0 Register (RTIUC0) [offset = 14h]**



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

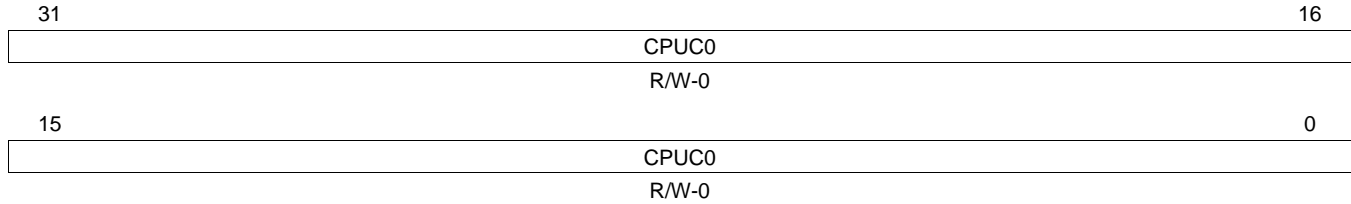
**Table 13-7. RTI Up Counter 0 Register (RTIUC0) Field Descriptions**

Bit	Field	Value	Description
31-0	UC0	0-FFFF FFFFh	<p>Up counter 0. This register holds the current value of the up counter 0 and prescales the RTI clock. It will be only updated by a previous read of free running counter 0 (RTIFRC0). This method of updating effectively gives a 64-bit read of both counters, without having the problem of a counter being updated between two consecutive reads on up counter 0 (RTIUC0) and free running counter 0 (RTIFRC0).</p> <p>A read of this counter returns the value of the counter at the time RTIFRC0 was read.</p> <p>A write to this counter presets it with a value. The counter then increments from this written value upwards.</p> <p>Note: If counters must be preset, they must be disabled in the RTIGCTRL register to ensure consistency between RTIUC0 and RTIFRC0.</p> <p><b>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.</b></p>

### 13.3.7 RTI Compare Up Counter 0 Register (RTICPUC0)

The compare up counter 0 register holds the value to be compared with prescale counter 0 (RTIUC0). This register is shown in [Figure 13-18](#) and described in [Table 13-8](#).

**Figure 13-18. RTI Compare Up Counter 0 Register (RTICPUC0) [offset = 18h]**



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

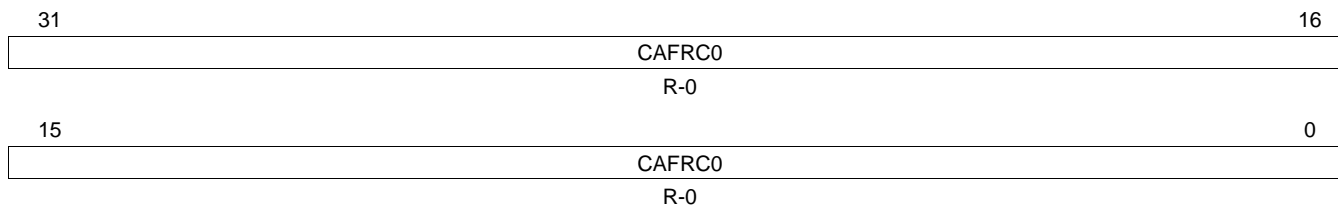
**Table 13-8. RTI Compare Up Counter 0 Register (RTICPUC0) Field Descriptions**

Bit	Field	Value	Description
31-0	CPUC0	0-FFFF FFFFh	<p>Compare up counter 0. This register holds the value that is compared with the up counter 0. When the compare shows a match, the free running counter 0 (RTIFRC0) is incremented. RTIUC0 is set to 0 when the counter value matches the RTICPUC0 value. The value set in this register prescales the RTI clock.</p> <p>If CPUC0 = 0, then  <math>f_{FRC0} = RTICLK/2^{32}</math></p> <p>If CPUC0 ≠ 0, then  <math>f_{FRC0} = RTICLK/(RTICPUC0+1)</math></p> <p>A read of this register returns the current compare value.</p> <p>A write to this register:</p> <ul style="list-style-type: none"> <li>• If TBEXT = 0, the compare value is updated.</li> <li>• If TBEXT = 1, the compare value is unchanged.</li> </ul>

### 13.3.8 RTI Capture Free Running Counter 0 Register (RTICAFRC0)

The capture free running counter 0 register holds the free running counter 0 on external events. This register is shown in [Figure 13-19](#) and described in [Table 13-9](#).

**Figure 13-19. RTI Capture Free Running Counter 0 Register (RTICAFRC0) [offset = 20h]**



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

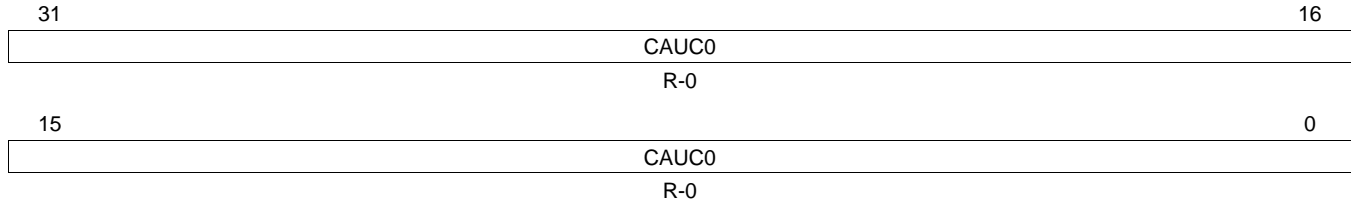
**Table 13-9. RTI Capture Free Running Counter 0 Register (RTICAFRC0) Field Descriptions**

Bit	Field	Value	Description
31-0	CAFRC0	0-FFFF FFFFh	Capture free running counter 0. This register captures the current value of the free running counter 0 (RTIFRC0) when an event occurs, controlled by the external capture control block. A read of this register returns the value of RTIFRC0 on a capture event.

### 13.3.9 RTI Capture Up Counter 0 Register (RTICAUC0)

The capture up counter 0 register holds the current value of prescale counter 0 on external events. This register is shown in [Figure 13-20](#) and described in [Table 13-10](#).

**Figure 13-20. RTI Capture Up Counter 0 Register (RTICAUC0) [offset = 24]**



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

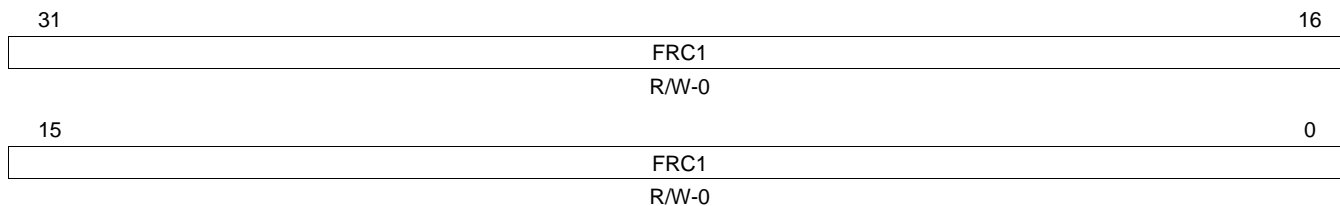
**Table 13-10. RTI Capture Up Counter 0 Register (RTICAUC0) Field Descriptions**

Bit	Field	Value	Description
31-0	CAUC0	0-FFFF FFFFh	<p>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.</p> <p><b>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.</b></p> <p>A read of this register returns the value of RTIUC0 on a capture event.</p>

### 13.3.10 RTI Free Running Counter 1 Register (RTIFRC1)

The free running counter 1 register holds the current value of the free running counter 1. This register is shown in [Figure 13-21](#) and described in [Table 13-11](#).

**Figure 13-21. RTI Free Running Counter 1 Register (RTIFRC1) [offset = 30h]**



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

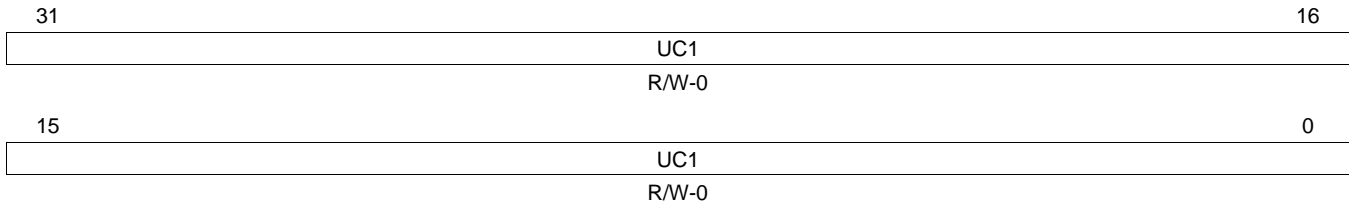
**Table 13-11. RTI Free Running Counter 1 Register (RTIFRC1) Field Descriptions**

Bit	Field	Value	Description
31-0	FRC1	0-FFFF FFFFh	<p>Free running counter 1. This register holds the current value of the free running counter 1 and will be updated continuously.</p> <p>A read of this register returns the current value of the counter.</p> <p>A write to this register presets the counter. The counter increments then from this written value upwards.</p> <p><b>Note: If counters must be preset, they must be disabled in the RTIGCTRL register to ensure consistency between RTIUC1 and RTIFRC1.</b></p>

### 13.3.11 RTI Up Counter 1 Register (RTIUC1)

The up counter 1 register holds the current value of the prescale counter 1. This register is shown in [Figure 13-22](#) and described in [Table 13-12](#).

**Figure 13-22. RTI Up Counter 1 Register (RTIUC1) [offset = 34h]**



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

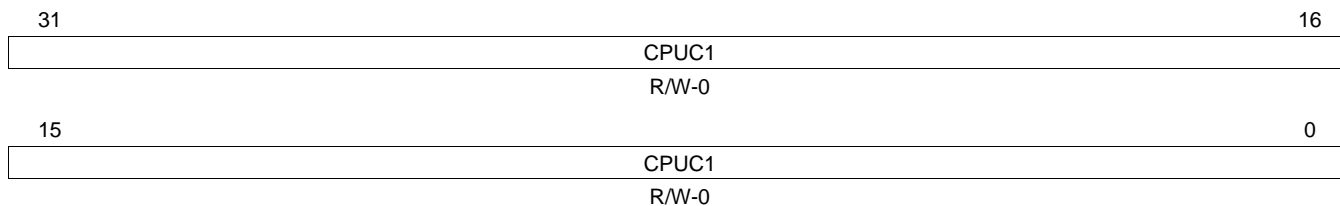
**Table 13-12. RTI Up Counter 1 Register (RTIUC1) Field Descriptions**

Bit	Field	Value	Description
31-0	UC1	0-FFFF FFFFh	<p>Up counter 1. This register holds the current value of the up counter 1 and prescales the RTI clock. It will be only updated by a previous read of free running counter 1 (RTIFRC1). This method of updating effectively gives a 64-bit read of both counters, without having the problem of a counter being updated between two consecutive reads on RTIUC1 and RTIFRC1.</p> <p>A read of this register will return the value of the counter when the RTIFRC1 was read.</p> <p>A write to this register presets the counter. The counter then increments from this written value upwards.</p> <p><b>Note: If counters must be preset, they must be disabled in the RTIGCTRL register to ensure consistency between RTIUC1 and RTIFRC1.</b></p> <p><b>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.</b></p>

### 13.3.12 RTI Compare Up Counter 1 Register (RTICPUC1)

The compare up counter 1 register holds the value compared with prescale counter 1. This register is shown in [Figure 13-23](#) and described in [Table 13-13](#).

**Figure 13-23. RTI Compare Up Counter 1 Register (RTICPUC1) [offset = 38h]**



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

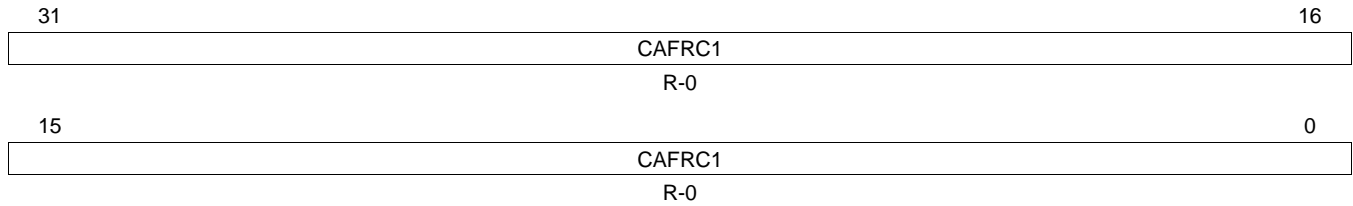
**Table 13-13. RTI Compare Up Counter 1 Register (RTICPUC1) Field Descriptions**

Bit	Field	Value	Description
31-0	CPUC1	0-FFFF FFFFh	<p>Compare up counter 1. This register holds the compare value, which is compared with the up counter 1. When the compare matches, the free running counter 1 (RTIFRC1) is incremented. The up counter is 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:</p> <p>If CPUC1 = 0, then  <math>f_{\text{FRC1}} = \text{RTICLK}/2^{32}</math></p> <p>If CPUC1 <math>\neq</math> 0, then  <math>f_{\text{FRC1}} = \text{RTICLK}/(\text{RTICPUC1}+1)</math></p> <p>A read of this register returns the current compare value.  A write to this register updates the compare value.</p>

**13.3.13 RTI Capture Free Running Counter 1 Register (RTICAFRC1)**

The capture free running counter 1 register holds the current value of free running counter 1 on external events. This register is shown in [Figure 13-24](#) and described in [Table 13-14](#).

**Figure 13-24. RTI Capture Free Running Counter 1 Register (RTICAFRC1) [offset = 40h]**



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

**Table 13-14. RTI Capture Free Running Counter 1 Register (RTICAFRC1) Field Descriptions**

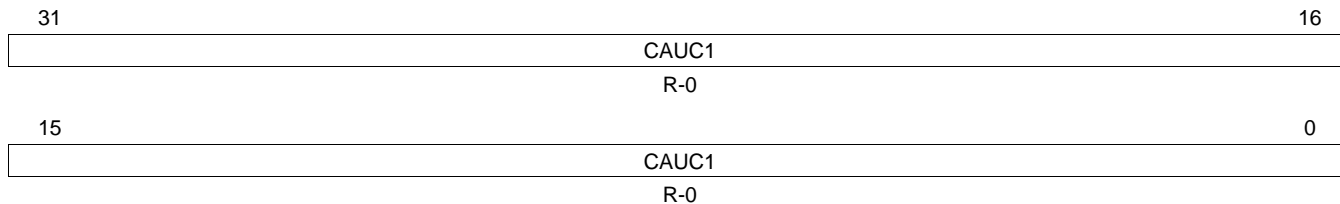
Bit	Field	Value	Description
31-0	CAFRC1	0-FFFF FFFFh	Capture free running counter 1. This register captures the current value of the free running counter 1 (RTIFRC1) when an event occurs, controlled by the external capture control block. A read of this register returns the value of RTIFRC1 on a capture event.



### 13.3.14 RTI Capture Up Counter 1 Register (RTICAUC1)

The capture up counter 1 register holds the current value of prescale counter 1 on external events. This register is shown in [Figure 13-25](#) and described in [Table 13-15](#).

**Figure 13-25. RTI Capture Up Counter 1 Register (RTICAUC1) [offset = 44h]**



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

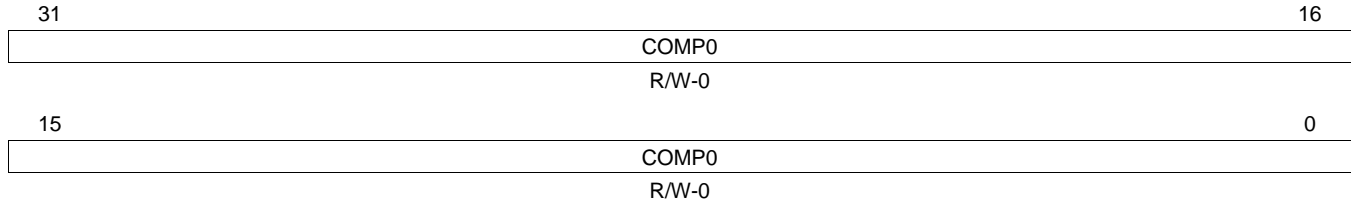
**Table 13-15. RTI Capture Up Counter 1 Register (RTICAUC1) Field Descriptions**

Bit	Field	Value	Description
31-0	CAUC1	0-FFFF FFFFh	<p>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.</p> <p><b>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.</b></p> <p>A read of this register returns the value of RTIUC1 on a capture event.</p>

### 13.3.15 RTI Compare 0 Register (RTICOMP0)

The compare 0 register holds the value to be compared with the counters. This register is shown in [Figure 13-26](#) and described in [Table 13-16](#).

**Figure 13-26. RTI Compare 0 Register (RTICOMP0) [offset = 50h]**



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

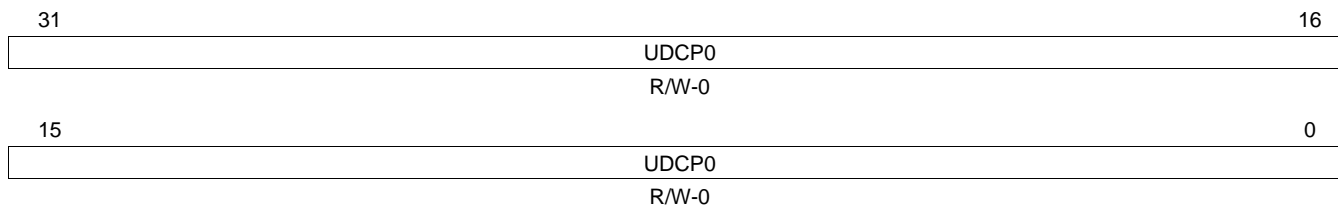
**Table 13-16. RTI Compare 0 Register (RTICOMP0) Field Descriptions**

Bit	Field	Value	Description
31-0	COMP0	0-FFFF FFFFh	<p>Compare 0. This registers holds a value that is compared with the counter selected in the compare control logic. If RTIFRC0 or RTIFRC1, depending on the counter selected, matches the compare value, an interrupt is flagged. With this register it is also possible to initiate a DMA request.</p> <p>A read of this register will return the current compare value.</p> <p>A write to this register (in privileged mode only) will update the compare register with a new compare value.</p>

### 13.3.16 RTI Update Compare 0 Register (RTIUDCP0)

The update compare 0 register holds the value to be added to the compare register 0 value on a compare match. This register is shown in [Figure 13-27](#) and described in [Table 13-17](#).

**Figure 13-27. RTI Update Compare 0 Register (RTIUDCP0) [offset = 54h]**



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

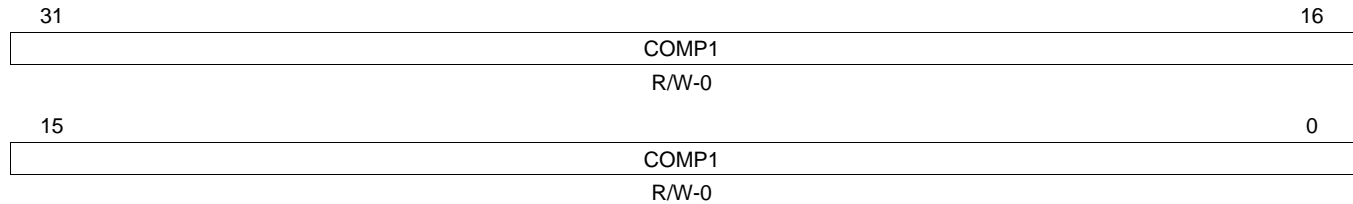
**Table 13-17. RTI Update Compare 0 Register (RTIUDCP0) Field Descriptions**

Bit	Field	Value	Description
31-0	UDCP0	0-FFFF FFFFh	<p>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.</p> <p>A read of this register will return the value to be added to the RTICOMP0 register on the next compare match.</p> <p>A write to this register will provide a new update value.</p>

### 13.3.17 RTI Compare 1 Register (RTICOMP1)

The compare 1 register holds the value to be compared to the counters. This register is shown in [Figure 13-28](#) and described in [Table 13-18](#).

**Figure 13-28. RTI Compare 1 Register (RTICOMP1) [offset = 58h]**



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

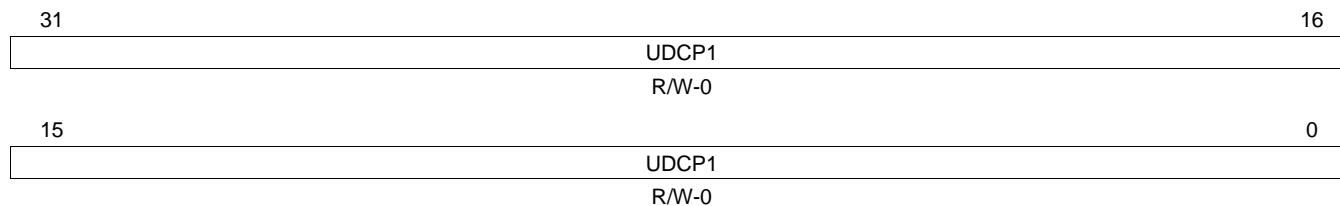
**Table 13-18. RTI Compare 1 Register (RTICOMP1) Field Descriptions**

Bit	Field	Value	Description
31-0	COMP1	0-FFFF FFFFh	<p>Compare 1. This register holds a value that is compared with the counter selected in the compare control logic. If RTIFRC0 or RTIFRC1, depending on the counter selected, matches this compare value, an interrupt is flagged. With this register, it is possible to initiate a DMA request.</p> <p>A read of this register will return the current compare value.</p> <p>A write to this register will update the compare register with a new compare value.</p>

### 13.3.18 RTI Update Compare 1 Register (RTIUDCP1)

The update compare 1 register holds the value to be added to the compare register 1 value on a compare match. This register is shown in [Figure 13-29](#) and described in [Table 13-19](#).

**Figure 13-29. RTI Update Compare 1 Register (RTIUDCP1) [offset = 5Ch]**



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

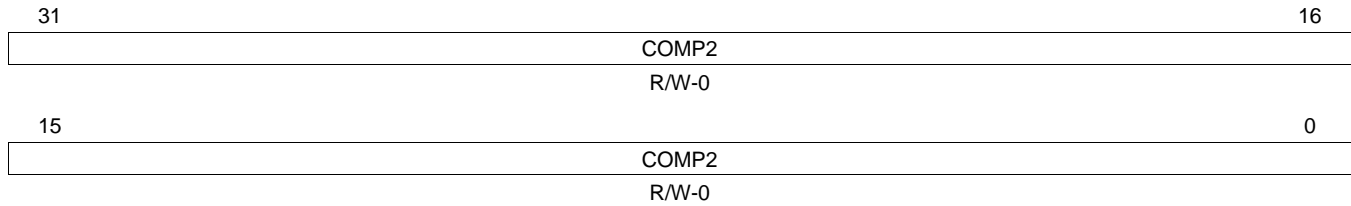
**Table 13-19. RTI Update Compare 1 Register (RTIUDCP1) Field Descriptions**

Bit	Field	Value	Description
31-0	UDCP1	0-FFFF FFFFh	<p>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.</p> <p>A read of this register will return the value to be added to the RTICOMP1 register on the next compare match. .</p> <p>A write to this register will provide a new update value</p>

### 13.3.19 RTI Compare 2 Register (RTICOMP2)

The compare 2 register holds the value to be compared to the counters. This register is shown in [Figure 13-30](#) and described in [Table 13-20](#).

**Figure 13-30. RTI Compare 2 Register (RTICOMP2) [offset = 60h]**



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

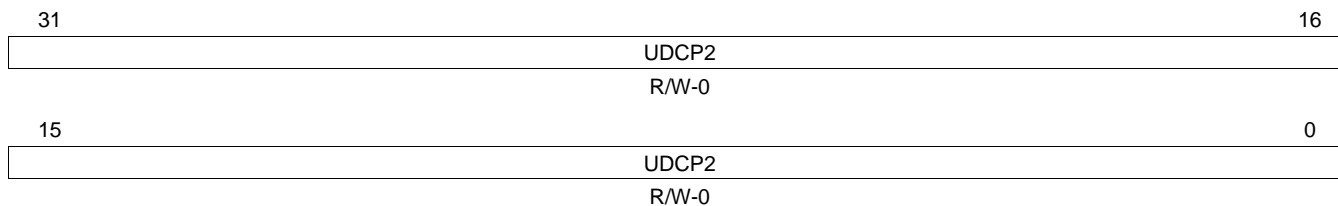
**Table 13-20. RTI Compare 2 Register (RTICOMP2) Field Descriptions**

Bit	Field	Value	Description
31-0	COMP2	0-FFFF FFFFh	<p>Compare 2. This register holds a value that is compared with the counter selected in the compare control logic. If RTIFRC0 or RTIFRC1, depending on the counter selected, matches this compare value, an interrupt is flagged. With this register, it is possible to initiate a DMA request.</p> <p>A read of this register will return the current compare value.</p> <p>A write to this register (in privileged mode only) will provide a new compare value.</p>

### 13.3.20 RTI Update Compare 2 Register (RTIUDCP2)

The update compare 2 register holds the value to be added to the compare register 2 value on a compare match. This register is shown in [Figure 13-31](#) and described in [Table 13-21](#).

**Figure 13-31. RTI Update Compare 2 Register (RTIUDCP2) [offset = 64h]**



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

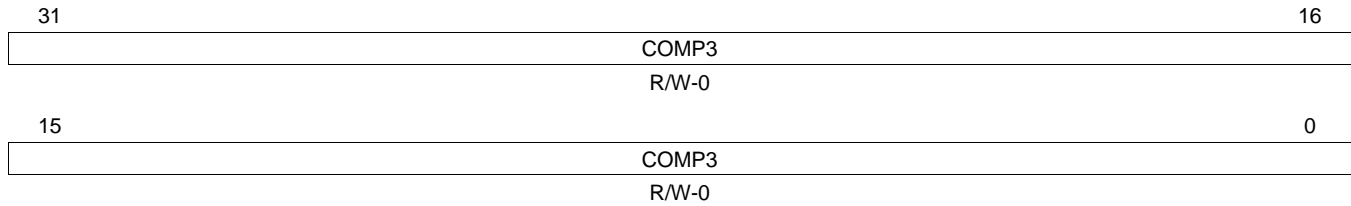
**Table 13-21. RTI Update Compare 2 Register (RTIUDCP2) Field Descriptions**

Bit	Field	Value	Description
31-0	UDCP2	0-FFFF FFFFh	<p>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.</p> <p>A read of this register will return the value to be added to the RTICOMP2 register on the next compare match.</p> <p>A write to this register will provide a new update value.</p>

### 13.3.21 RTI Compare 3 Register (RTICOMP3)

The compare 3 register holds the value to be compared to the counters. This register is shown in [Figure 13-32](#) and described in [Table 13-22](#).

**Figure 13-32. RTI Compare 3 Register (RTICOMP3) [offset = 68h]**



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

**Table 13-22. RTI Compare 3 Register (RTICOMP3) Field Descriptions**

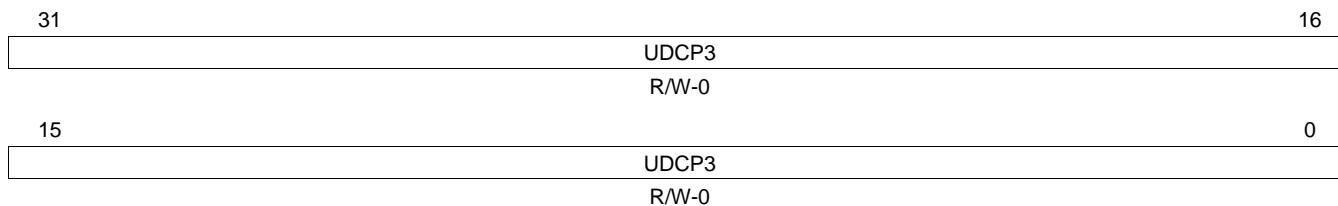
Bit	Field	Value	Description
31-0	COMP3	0-FFFF FFFFh	<p>Compare 3. This register holds a value that is compared with the counter selected in the compare control logic. If RTIFRC0 or RTIFRC1, depending on the counter selected, matches this compare value, an interrupt is flagged. With this register, it is possible to initiate a DMA request.</p> <p>A read of this register will return the current compare value.</p> <p>A write to this register will provide a new compare value.</p>



### 13.3.22 RTI Update Compare 3 Register (RTIUDCP3)

The update compare 3 register holds the value to be added to the compare register 3 value on a compare match. This register is shown in [Figure 13-33](#) and described in [Table 13-23](#).

**Figure 13-33. RTI Update Compare 3 Register (RTIUDCP3) [offset = 6Ch]**



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

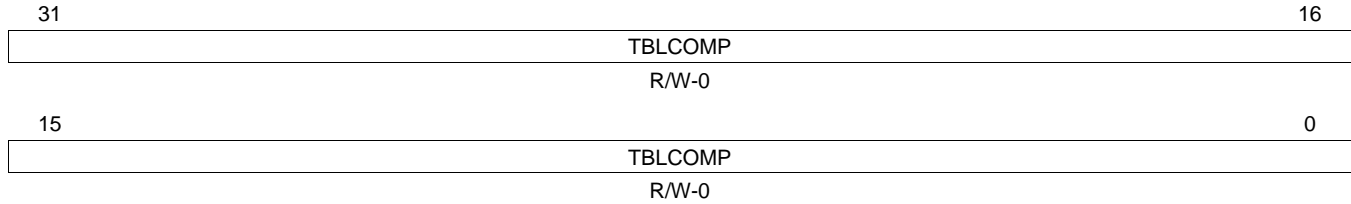
**Table 13-23. RTI Update Compare 3 Register (RTIUDCP3) Field Descriptions**

Bit	Field	Value	Description
31-0	UDCP3	0-FFFF FFFFh	<p>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.</p> <p>A read of this register will return the value to be added to the RTICOMP3 register on the next compare match.</p> <p>A write to this register will provide a new update value.</p>

### 13.3.23 RTI Timebase Low Compare Register (RTITBLCOMP)

The timebase low compare register holds the value to activate the edge detection circuit. This register is shown in [Figure 13-34](#) and described in [Table 13-24](#).

**Figure 13-34. RTI Timebase Low Compare Register (RTITBLCOMP) [offset = 70h]**



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

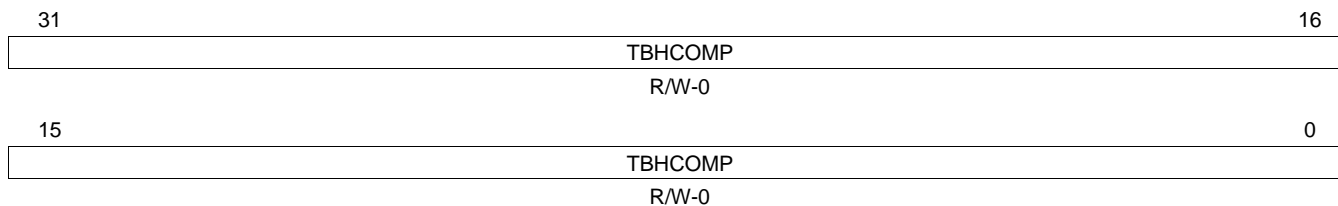
**Table 13-24. RTI Timebase Low Compare Register (RTITBLCOMP) Field Descriptions**

Bit	Field	Value	Description
31-0	TBLCOMP	0-FFFF FFFFh	Timebase low compare value. This value determines when the edge detection circuit starts monitoring the NTU signal. It will be compared with RTIUC0. A read of this register will return the current compare value. A write to this register has the following effects: If TBEXT = 0: The compare value is updated. If TBEXT = 1: The compare value is not changed.

### 13.3.24 RTI Timebase High Compare Register (RTITBHCMP)

The timebase high compare register holds the value to deactivate the edge detection circuit. This register is shown in [Figure 13-35](#) and described in [Table 13-25](#).

**Figure 13-35. RTI Timebase High Compare Register (RTITBHCMP) [offset = 74h]**



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

**Table 13-25. RTI Timebase High Compare Register (RTITBHCMP) Field Descriptions**

Bit	Field	Value	Description
31-0	TBHCOMP	0-FFFF FFFFh	<p>Timebase high compare value. This value determines when the edge detection circuit will stop monitoring the NTU signal. It will be compared with RTIUC0.</p> <p>RTITBHCMP must be less than RTICPUC0 because RTIUC0 will be reset when RTICPUC0 is reached.</p> <p>Example: The NTU edge detection circuit should be active <math>\pm 10</math> RTICLK cycles around RTICPUC0.</p> <ul style="list-style-type: none"> <li>• RTICPUC0 = 0050h</li> <li>• RTITBLCMP = 0046h</li> <li>• RTITBHCMP = 0009h</li> </ul> <p>A read of this register will return the current compare value.</p> <p>A write to this register has the following effects:            If TBEXT = 0: The compare value is updated.            If TBEXT = 1: The compare value is not changed.</p>

### 13.3.25 RTI Set Interrupt Enable Register (RTISETINTENA)

This register prevents the necessity of a read-modify-write operation if a particular interrupt should be enabled. This register is shown in [Figure 13-36](#) and described in [Table 13-26](#).

**Figure 13-36. RTI Set Interrupt Control Register (RTISETINTENA) [offset = 80h]**

31	Reserved				24	
R-0						
23	Reserved			18	17	16
R-0			SETOVL1INT	SETOVL0INT	SETTBINT	
R-0			R/W-0	R/W-0	R/W-0	
15	Reserved		11	10	9	8
R-0		SETDMA3	SETDMA2	SETDMA1	SETDMA0	
R-0		R/W-0	R/W-0	R/W-0	R/W-0	
7	Reserved		3	2	1	0
R-0		SETINT3	SETINT2	SETINT1	SETINT0	
R-0		R/W-0	R/W-0	R/W-0	R/W-0	

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

**Table 13-26. RTI Set Interrupt Control Register (RTISETINTENA) Field Descriptions**

Bit	Field	Value	Description
31-19	Reserved	0	Read returns 0. Writes have no effect.
18	SETOVL1INT	0	Set free running counter 1 overflow interrupt. <i>Read:</i> The interrupt is disabled. <i>Write:</i> The corresponding bit is unchanged.
		1	<i>Read or write:</i> The interrupt is enabled.
17	SETOVL0INT	0	Set free running counter 0 overflow interrupt. <i>Read:</i> The interrupt is disabled. <i>Write:</i> The corresponding bit is unchanged.
		1	<i>Read or write:</i> The interrupt is enabled.
16	SETTBINT	0	Set timebase interrupt. <i>Read:</i> The interrupt is disabled. <i>Write:</i> The corresponding bit is unchanged.
		1	<i>Read or write:</i> The interrupt is enabled.
15-12	Reserved	0	Read returns 0. Writes have no effect.
11	SETDMA3	0	Set compare DMA request 3. <i>Read:</i> The DMA request is disabled. <i>Write:</i> The DMA request is unchanged.
		1	<i>Read or write:</i> The DMA request is enabled.
10	SETDMA2	0	Set compare DMA request 2. <i>Read:</i> The DMA request is disabled. <i>Write:</i> The DMA request is unchanged.
		1	<i>Read or write:</i> The DMA request is enabled.
9	SETDMA1	0	Set compare DMA request 1. <i>Read:</i> The DMA request is disabled. <i>Write:</i> The DMA request is unchanged.
		1	<i>Read or write:</i> The DMA request is enabled.

**Table 13-26. RTI Set Interrupt Control Register (RTISETINTENA) Field Descriptions (continued)**

Bit	Field	Value	Description
8	SETDMA0	0	Set compare DMA request 0. <i>Read:</i> The DMA request is disabled. <i>Write:</i> The DMA request is unchanged.
		1	<i>Read or write:</i> The DMA request is enabled.
7-4	Reserved	0	Read returns 0. Writes have no effect.
3	SETINT3	0	Set compare interrupt 3. <i>Read:</i> The interrupt is disabled. <i>Write:</i> The corresponding bit is unchanged.
		1	<i>Read or write:</i> The interrupt is enabled.
2	SETINT2	0	Set compare interrupt 2. <i>Read:</i> The interrupt is disabled. <i>Write:</i> The corresponding bit is unchanged.
		1	<i>Read or write:</i> The interrupt is enabled.
1	SETINT1	0	Set compare interrupt 1. <i>Read:</i> The interrupt is disabled. <i>Write:</i> The corresponding bit is unchanged.
		1	<i>Read or write:</i> The interrupt is enabled.
0	SETINT0	0	Set compare interrupt 0. <i>Read:</i> The interrupt is disabled. <i>Write:</i> The corresponding bit is unchanged.
		1	<i>Read or write:</i> The interrupt is enabled.

### 13.3.26 RTI Clear Interrupt Enable Register (RTICLEARINTENA)

This register prevents the necessity of a read-modify-write operation if a particular interrupt should be disabled. This register is shown in [Figure 13-37](#) and described in [Table 13-27](#).

**Figure 13-37. RTI Clear Interrupt Control Register (RTICLEARINTENA) [offset = 84h]**

31	Reserved				24
R-0					
23	Reserved		18	17	16
R-0			R/W-0	R/W-0	R/W-0
15	Reserved		10	9	8
R-0			R/W-0	R/W-0	R/W-0
7	Reserved		2	1	0
R-0			R/W-0	R/W-0	R/W-0

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

**Table 13-27. RTI Clear Interrupt Control Register (RTICLEARINTENA) Field Descriptions**

Bit	Field	Value	Description
31-19	Reserved	0	Read returns 0. Writes have no effect.
18	CLEAROVL1INT	0 1	Clear free running counter 1 overflow interrupt. <i>Read:</i> The interrupt is disabled. <i>Write:</i> The corresponding bit is unchanged. <i>Read:</i> The interrupt is enabled. <i>Write:</i> The interrupt is disabled.
17	CLEAROVL0INT	0 1	Clear free running counter 0 overflow interrupt. <i>Read:</i> The interrupt is disabled. <i>Write:</i> The corresponding bit is unchanged. <i>Read:</i> The interrupt is enabled. <i>Write:</i> The interrupt is disabled.
16	CLEARTBINT	0 1	Clear timebase interrupt. <i>Read:</i> The interrupt is disabled. <i>Write:</i> The corresponding bit is unchanged. <i>Read:</i> The interrupt is enabled. <i>Write:</i> The interrupt is disabled.
15-12	Reserved	0	Read returns 0. Writes have no effect.
11	CLEARDMA3	0 1	Clear compare DMA request 3. <i>Read:</i> The DMA request is disabled. <i>Write:</i> The corresponding bit is unchanged. <i>Read:</i> The DMA request is enabled. <i>Write:</i> The DMA request is disabled.
10	CLEARDMA2	0 1	Clear compare DMA request 2. <i>Read:</i> The DMA request is disabled. <i>Write:</i> The corresponding bit is unchanged. <i>Read:</i> The DMA request is enabled. <i>Write:</i> The DMA request is disabled.

**Table 13-27. RTI Clear Interrupt Control Register (RTICLEARINTENA) Field Descriptions (continued)**

Bit	Field	Value	Description
9	CLEARDMA1	0	Clear compare DMA request 1. <i>Read:</i> The DMA request is disabled. <i>Write:</i> The corresponding bit is unchanged.
		1	<i>Read:</i> The DMA request is enabled. <i>Write:</i> The DMA request is disabled.
8	CLEARDMA1	0	Clear compare DMA request 0. <i>Read:</i> The DMA request is disabled. <i>Write:</i> The corresponding bit is unchanged.
		1	<i>Read:</i> The DMA request is enabled. <i>Write:</i> The DMA request is disabled.
7-4	Reserved	0	Read returns 0. Writes have no effect.
3	CLEARINT3	0	Clear compare interrupt 3. <i>Read:</i> The interrupt is disabled. <i>Write:</i> The corresponding bit is unchanged.
		1	<i>Read:</i> The interrupt is enabled. <i>Write:</i> The interrupt is disabled.
2	CLEARINT2	0	Clear compare interrupt 2. <i>Read:</i> The interrupt is disabled. <i>Write:</i> The corresponding bit is unchanged.
		1	<i>Read:</i> The interrupt is enabled. <i>Write:</i> The interrupt is disabled.
1	CLEARINT1	0	Clear compare interrupt 1. <i>Read:</i> The interrupt is disabled. <i>Write:</i> The corresponding bit is unchanged.
		1	<i>Read:</i> The interrupt is enabled. <i>Write:</i> The interrupt is disabled.
0	CLEARINT0	0	Clear compare interrupt 0. <i>Read:</i> The interrupt is disabled. <i>Write:</i> The corresponding bit is unchanged.
		1	<i>Read:</i> The interrupt is enabled. <i>Write:</i> The interrupt is disabled.

### 13.3.27 RTI Interrupt Flag Register (RTIINTFLAG)

The corresponding flags are set at every compare match of the RTIFRCx and RTICOMPx values, whether the interrupt is enabled or not. This register is shown in [Figure 13-38](#) and described in [Table 13-28](#).

**Figure 13-38. RTI Interrupt Flag Register (RTIINTFLAG) [offset = 88h]**

31	Reserved	19	18	17	16
			OVL1INT	OVL0INT	TBINT
	R-0		R/W-0	R/W-0	R/W-0
15	Reserved	4	3	2	1
			INT3	INT2	INT1
	R-0		R/W-0	R/W-0	R/W-0
			INT0		

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

**Table 13-28. RTI Interrupt Flag Register (RTIINTFLAG) Field Descriptions**

Bit	Field	Value	Description
31-19	Reserved	0	Read returns 0. Writes have no effect.
18	OVL1INT	0	Free running counter 1 overflow interrupt flag. This bit determines if an interrupt is pending. <i>Read:</i> No interrupt is pending. <i>Write:</i> The bit is unchanged.
		1	<i>Read:</i> An interrupt is pending. <i>Write:</i> The bit is set to 0.
17	OVL0INT	0	Free running counter 0 overflow interrupt flag. This bit determines if an interrupt is pending. <i>Read:</i> No interrupt is pending. <i>Write:</i> The bit is unchanged.
		1	<i>Read:</i> An interrupt is pending. <i>Write:</i> The bit is set to 0.
16	TBINT	0	Timebase interrupt flag. This flag is set when the TBEXT bit is cleared by detection of a missing external clock edge. It will not be set by clearing TBEXT by software. It determines if an interrupt is pending. <i>Read:</i> No interrupt is pending. <i>Write:</i> The bit is unchanged.
		1	<i>Read:</i> An interrupt is pending. <i>Write:</i> The bit is set to 0.
15-4	Reserved	0	Read returns 0. Writes have no effect.
3	INT3	0	Interrupt flag 3. These bits determine if an interrupt due to a Compare 3 match is pending. <i>Read:</i> No interrupt is pending. <i>Write:</i> The bit is unchanged.
		1	<i>Read:</i> An interrupt is pending. <i>Write:</i> The bit is set to 0.
2	INT2	0	Interrupt flag 2. These bits determine if an interrupt due to a Compare 2 match is pending. <i>Read:</i> No interrupt is pending. <i>Write:</i> The bit is unchanged.
		1	<i>Read:</i> An interrupt is pending. <i>Write:</i> The bit is set to 0.
1	INT1	0	Interrupt flag 1. These bits determine if an interrupt due to a Compare 1 match is pending. <i>Read:</i> No interrupt is pending. <i>Write:</i> The bit is unchanged.
		1	<i>Read:</i> An interrupt is pending. <i>Write:</i> The bit is set to 0.



**Table 13-28. RTI Interrupt Flag Register (RTIINTFLAG) Field Descriptions (continued)**

Bit	Field	Value	Description
0	INT0	0	Interrupt flag 0. These bits determine if an interrupt due to a Compare 0 match is pending. <i>Read:</i> No interrupt is pending. <i>Write:</i> The bit is unchanged.
		1	<i>Read:</i> An interrupt is pending. <i>Write:</i> The bit is set to 0.

### 13.3.28 Digital Watchdog Control Register (RTIDWDCTRL)

The software has to write to the DWDCTRL field in order to enable the DWD, as described below.

Once enabled, the watchdog can only be disabled by a system reset. The application cannot disable the watchdog.

This register is shown in [Figure 13-38](#) and described in [Table 13-28](#).

**Figure 13-39. Digital Watchdog Control Register (RTIDWDCTRL) [offset = 90h]**

31	DWDCTRL	16
	R/W-5312h	
15	DWDCTRL	0
	R/W-ACEDh	

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

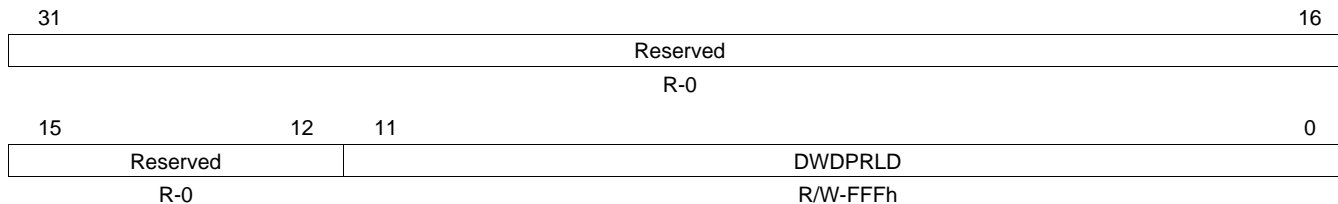
**Table 13-29. Digital Watchdog Control Register (RTIDWDCTRL) Field Descriptions**

Bit	Field	Value	Description
31-0	DWDCTRL	5312ACEDh	Digital Watchdog Control. <i>Read:</i> DWD counter is disabled. <i>Write:</i> State of DWD counter is unchanged (stays enabled or disabled).
		A98559DAh	<i>Read:</i> DWD counter is enabled. <i>Write:</i> DWD counter is enabled
		Any other value	<i>Read:</i> DWD counter state is unchanged (enabled or disabled). <i>Write:</i> State of DWD counter is unchanged (stays enabled or disabled). <b>Note:</b> Once the enable value is written, all other future writes are blocked. In other words, once DWD is enabled, it can only be disabled by system reset or power on reset.

### 13.3.29 Digital Watchdog Preload Register (RTIDWDPRLD)

This register sets the expiration time of the DWD. This register is shown in [Figure 13-38](#) and described in [Table 13-28](#).

**Figure 13-40. Digital Watchdog Preload Register (RTIDWDPRLD) [offset = 94h]**



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

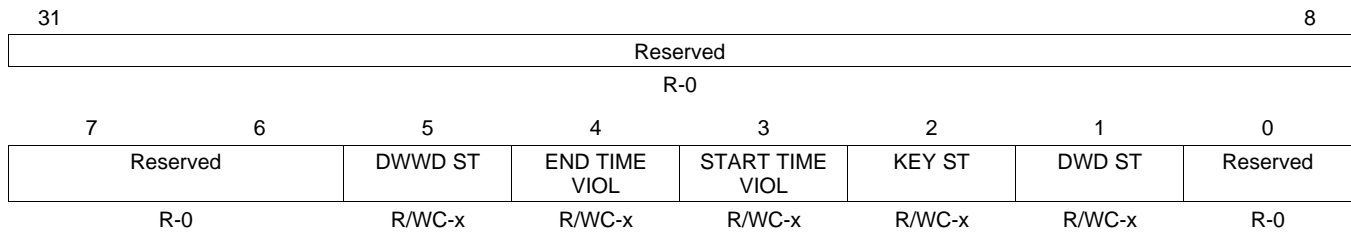
**Table 13-30. Digital Watchdog Preload Register (RTIDWDPRLD) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved		Reads return 0 and writes have no effect.
15-0	DWDPRLD	0-FFFh	Digital Watchdog Preload Value. <i>Read:</i> The current preload value. <i>Write:</i> Set the preload value. The DWD preload register can be configured only when the DWD is disabled. Therefore, the application can only configure the DWD preload register before it enables the DWD down counter.  The expiration time of the DWD Down Counter can be determined with following equation: $\text{texp} = (\text{DWDPRLD} + 1) \times 2^{13} / \text{RTICK1}$ where: DWDPRLD = 0...4095

### 13.3.30 Watchdog Status Register (RTIWDSTATUS)

This register records the status of the DWD. The values of the following status bits will not be affected by a soft reset. These bits are cleared by a power up reset, or by a write of '1'. These bits can be used for debug purposes. This register is shown in [Figure 13-38](#) and described in [Table 13-28](#).

**Figure 13-41. Watchdog Status Register (RTIWDSTATUS) [offset = 98h]**



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

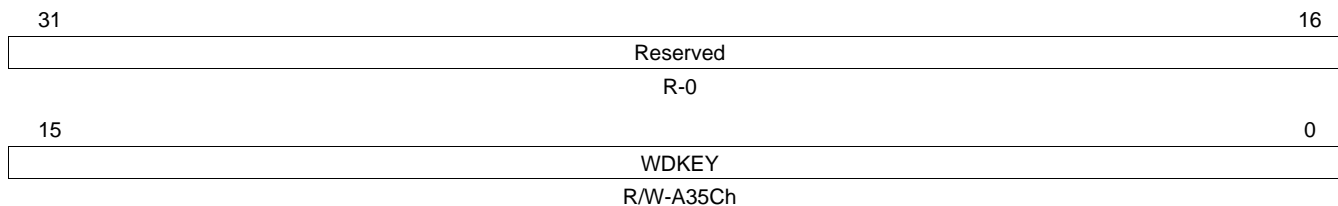
**Table 13-31. Watchdog Status Register (RTIWDSTATUS) Field Descriptions**

Bit	Field	Value	Description
31-6	Reserved	0	Read returns 0. Writes have no effect.
5	DWWWD ST	0	Windowed Watchdog Status <i>Read:</i> no time-window violation has occurred. <i>Write:</i> leaves the current value unchanged.
		1	<i>Read:</i> a time-window violation has occurred. The watchdog has generated either a system reset or a non-maskable interrupt to the CPU in this case. <i>Write:</i> clears the bit to 0. This will also clear all other status flags in the RTIWDSTATUS register. Clearing of the status flags will deassert the non-maskable interrupt generated due to violation of the DWWWD.
4	END TIME VIOL	0	Windowed Watchdog End Time Violation Status. This bit indicates whether the Watchdog counter expired. <i>Read:</i> no end-time window violation has occurred. <i>Write:</i> leaves the current value unchanged.
		1	<i>Read:</i> the end-time defined by the windowed watchdog configuration has been violated. <i>Write:</i> clears the bit to 0.
3	START TIME VIOL	0	Windowed Watchdog Start Time Violation Status. This bit indicates whether the key is written before the watchdog window opened up. <i>Read:</i> No start-time window violation has occurred. <i>Write:</i> Leaves the current value unchanged.
		1	<i>Read:</i> The start-time defined by the windowed watchdog configuration has been violated. <i>Write:</i> Clears the bit to 0.
2	KEY ST	0	Watchdog key status. This bit indicates a reset or NMI generated by a wrong key or key sequence written to the RTIWDKEY register. <i>Read:</i> No wrong key or key-sequence written. <i>Write:</i> The bit is unchanged.
		1	<i>Read:</i> Wrong key or key-sequence written to RTIWDKEY register. <i>Write:</i> The bit is set to 0.
1	DWD ST	0	DWD status. This bit is equivalent to bit END TIME VIOL. <i>Read:</i> No reset or NMI was generated. <i>Write:</i> The bit is unchanged.
		1	<i>Read:</i> A reset or NMI was generated. <i>Write:</i> The bit is set to 0.
0	Reserved	0	Read returns 0. Writes have no effect.

### 13.3.31 RTI Watchdog Key Register (RTIWDKEY)

This register must be written with the correct written key values to serve the watchdog. This register is shown in [Figure 13-42](#) and described in [Table 13-32](#).

**Figure 13-42. RTI Watchdog Key Register (RTIWDKEY) [offset = 9Ch]**



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

**Table 13-32. RTI Watchdog Key Register (RTIWDKEY) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved		Reads return 0 and writes have no effect.
15-0	WDKEY	0-FFFFh	Watchdog key. These bits provide the key sequence location. Reads returns the current WDKEY value. A write of 0xE51A followed by 0xA35C in two separate write operations defines the key sequence and reloads the DWD. Writing any other value causes a reset or NMI, as shown in <a href="#">Table 13-33</a> . Writing any other value will cause the WDKEY to reset to 0xA35C.

**NOTE:** It has to be taken into account that the write to the RTIWDKEY register takes 3 VCLK cycles.

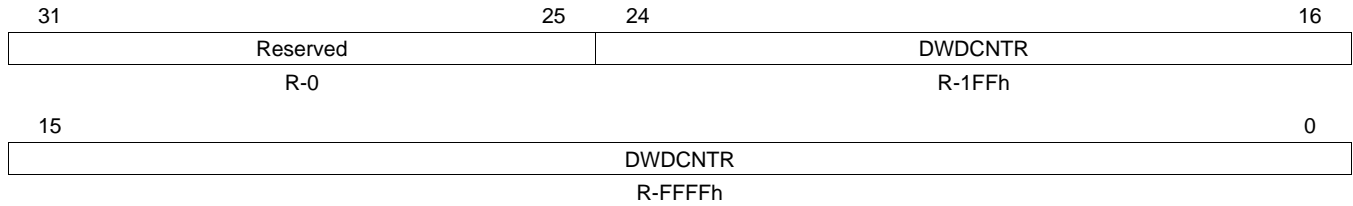
**Table 13-33. Example of a WDKEY Sequence**

Step	Value Written to WDKEY	Result
1	0x0A35C	No action
2	0x0A35C	No action
3	0x0E51A	WDKEY is enabled for reset or NMI by next 0x0A35C.
4	0x0E51A	WDKEY is enabled for reset or NMI by next 0x0A35C.
5	0x0E51A	WDKEY is enabled for reset or NMI by next 0x0A35C.
6	0x0A35C	Watchdog is reset.
7	0x0A35C	No action
8	0x0E51A	WDKEY is enabled for reset or NMI by next 0x0A35C.
9	0x0A35C	Watchdog is reset.
10	0x0E51A	WDKEY is enabled for reset or NMI by next 0x0A35C.
11	0x02345	System reset or NMI; incorrect value written to WDKEY.

### 13.3.32 RTI Digital Watchdog Down Counter (RTIDWDCNTR)

This register provides the current value of the DWD down counter. This register is shown in [Figure 13-43](#) and described in [Table 13-34](#).

**Figure 13-43. RTI Watchdog Down Counter Register (RTIDWDCNTR) [offset = A0h]**



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

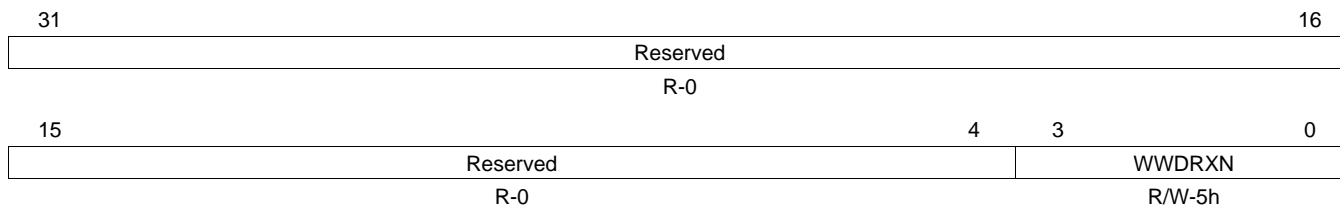
**Table 13-34. RTI Watchdog Down Counter Register (RTIDWDCNTR) Field Descriptions**

Bit	Field	Value	Description
31-25	Reserved		Reads return 0 and writes have no effect.
24-0	DWDNTR	0-1FF FFFFh	DWD down counter. Reads return the current counter value.

### 13.3.33 Digital Windowed Watchdog Reaction Control (RTIWWDRXNCTRL)

This register selects the DWWD reaction if the watchdog is serviced outside the time window. This register is shown in [Figure 13-44](#) and described in [Table 13-35](#).

**Figure 13-44. Digital Windowed Watchdog Reaction Control (RTIWWDRXNCTRL) [offset = A4h]**



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

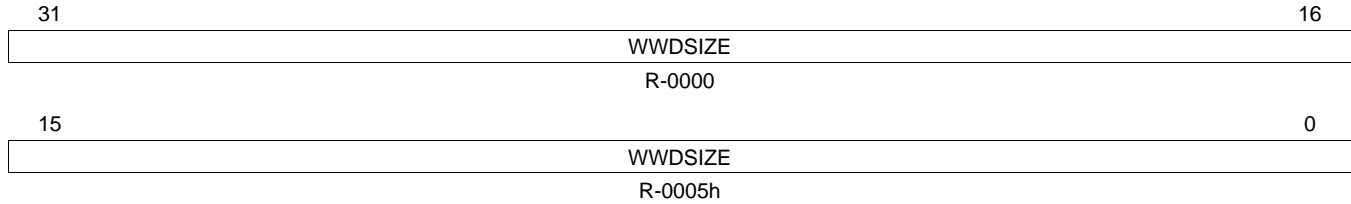
**Table 13-35. Digital Windowed Watchdog Reaction Control (RTIWWDRXNCTRL) Field Descriptions**

Bit	Field	Value	Description
31-4	Reserved		Reads return 0 and writes have no effect.
3-0	WWDRXN	5h	The windowed watchdog will cause a reset if the watchdog is serviced outside the time window defined by the configuration, or if the watchdog is not serviced at all.
		Ah	The windowed watchdog will generate a non-maskable interrupt to the CPU if the watchdog is serviced outside the time window defined by the configuration, or if the watchdog is not serviced at all.
		Any other value	The windowed watchdog will cause a reset if the watchdog is serviced outside the time window defined by the configuration, or if the watchdog is not serviced at all.  <b>Note:</b> The DWWD reaction can be selected by the application even when the DWWD counter is already enabled. If a change to the WWDRXN is made before the watchdog service window is opened, then the change in the configuration takes effect immediately. If a change to the WWDRXN is made when the watchdog service window is already open, then the change in configuration takes effect only after the watchdog is serviced.

### 13.3.34 Digital Windowed Watchdog Window Size Control (RTIWWDSIZECTRL)

This register selects the DWWD window size. This register is shown in [Figure 13-45](#) and described in [Table 13-36](#).

**Figure 13-45. Digital Windowed Watchdog Window Size Control (RTIWWDSIZECTRL) [offset = A8h]**



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

**Table 13-36. Digital Windowed Watchdog Window Size Control (RTIWWDSIZECTRL) Field Descriptions**

Bit	Field	Value	Description
31-0	WWDSIZE		The DWWD window size
		0000 0005h	100% (The functionality is the same as the standard time-out digital watchdog.)
		0000 0050h	50%
		0000 0500h	25%
		0000 5000h	12.5%
		0005 0000h	6.25%
		Any other value	3.125%
			<b>Note:</b> The DWWD window size can be selected by the application even when the DWWD counter is already enabled. If a change to the WWDSIZE is made before the watchdog service window is opened, then the change in the configuration takes effect immediately. If a change to the WWDSIZE is made when the watchdog service window is already open, then the change in configuration takes effect only after the watchdog is serviced.





## ***Cyclic Redundancy Check Controller (CRC) Module***

---

---

This chapter describes the cyclic redundancy check controller (CRC).

<b>Topic</b>	<b>Page</b>
<b>14.1 Overview</b> .....	<b>512</b>
<b>14.2 Module Operation</b> .....	<b>515</b>
<b>14.3 Example</b> .....	<b>526</b>
<b>14.4 CRC Control Registers</b> .....	<b>529</b>

## 14.1 Overview

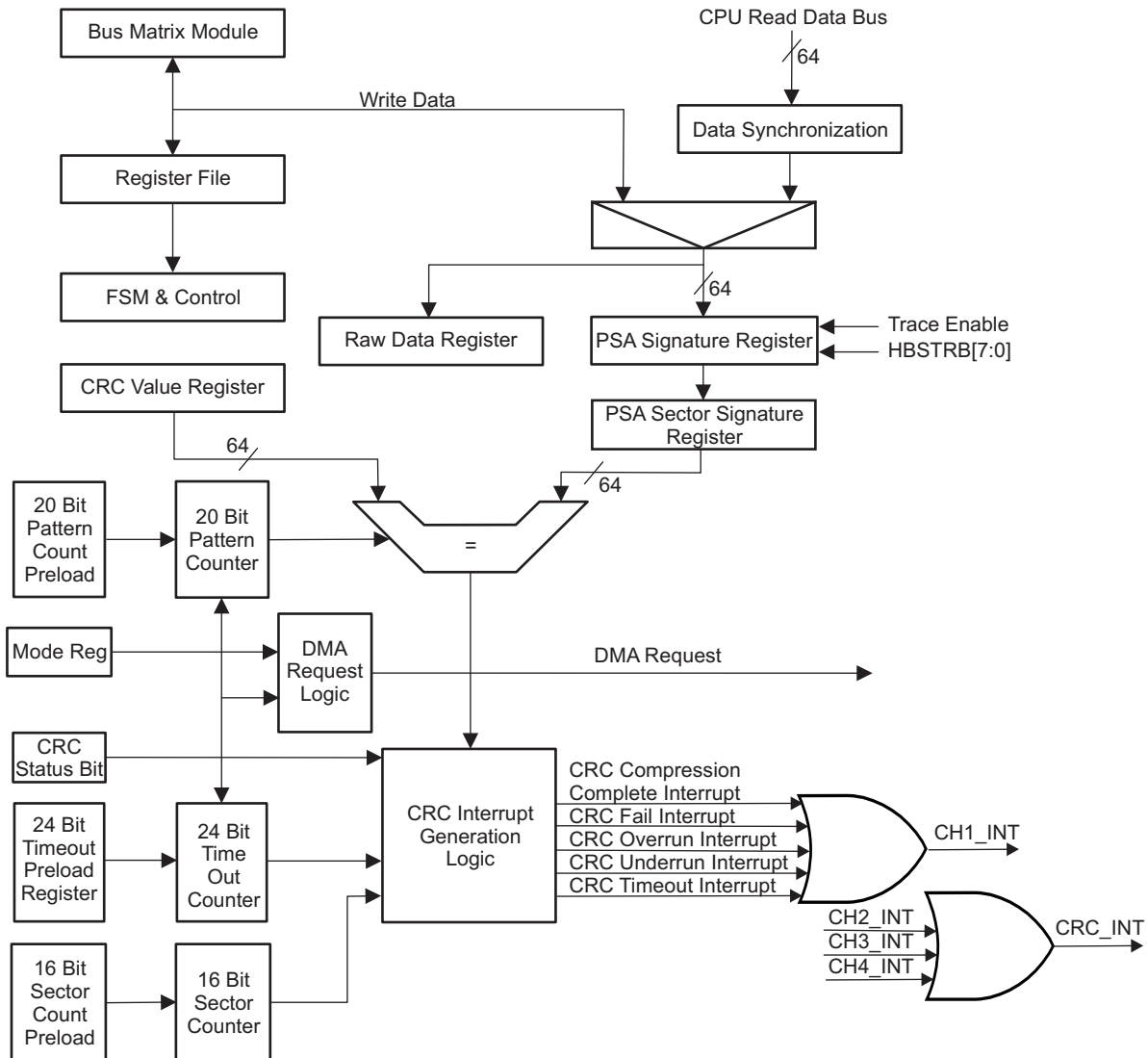
MCRC Controller is a module which is used to perform CRC (Cyclic Redundancy Check) to verify the integrity of memory system. A signature representing the contents of the memory is obtained when the contents of the memory are read into MCRC Controller. The responsibility of MCRC controller is to calculate the signature for a set of data and then compare the calculated signature value against a pre-determined good signature value. MCRC controller provides up to four channels to perform CRC calculation on multiple memories in parallel and can be used on any memory system. Channel 1 can also be put into data trace mode. In data trace mode, MCRC controller compresses each data being read through CPU read data bus.

### 14.1.1 Features

The CRC (Cyclic Redundancy Check) controller offers:

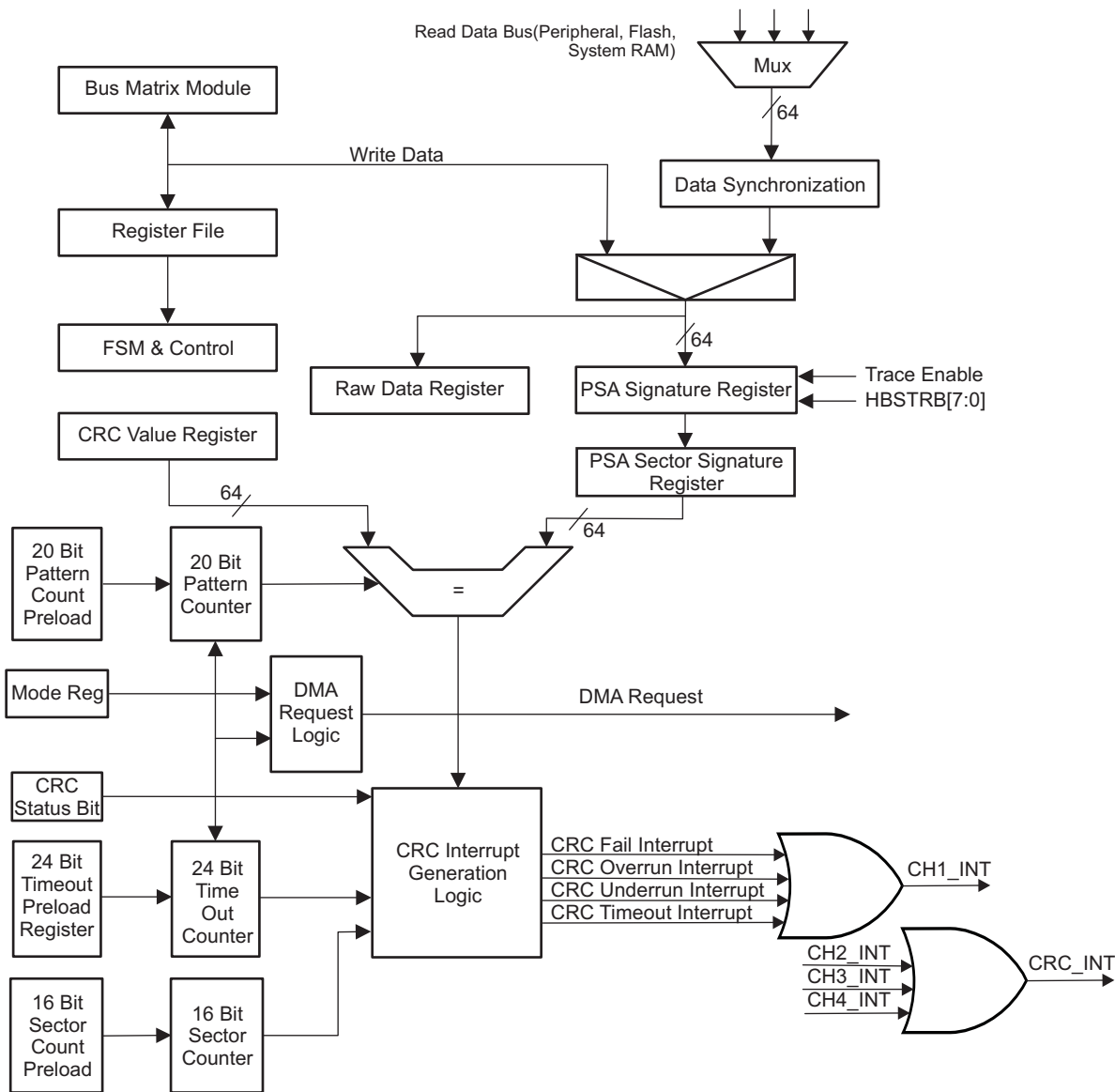
- Up to Four channels to perform background signature verification on any memory sub-system. This device supports two channels.
- Data compression on 8, 16, 32, and 64 bit data size.
- Maximum-length PSA (Parallel Signature Analysis) register constructed based on 64 bit primitive polynomial.
- Each channel has a CRC Value Register which contains the pre-determined CRC value.
- Use timed base event trigger from timer to initiate DMA data transfer.
- Programmable 20-bit pattern counter per channel to count the number of data patterns for compression.
- Three Three modes of operation. Auto, Semi-CPU Semi-CPU and Full-CPU.
- For each channel, CRC can be performed either by MCRC Controller or by CPU.
- Automatically perform signature verification without CPU intervention in AUTO mode.
- Generate interrupt to CPU in Semi-CPU mode to allow CPU to perform signature verification itself.
- Generate interrupt to CPU in Semi-CPU mode to allow CPU to perform signature verification itself.
- Generate CRC fail interrupt in AUTO mode if signature verification fails.
- Generate Timeout interrupt if CRC is not performed within the time limit.
- Generate DMA request per channel to initiate CRC value transfer.
- Peripheral Bus slave interface.
- Data trace capability on Peripheral Bus Master, Flash and System RAM data buses.

Figure 14-1. CRC Controller Block Diagram For One Channel



**NOTE:** Only Channel 1 can support data trace.

**Figure 14-2. MCRC Controller Block Diagram For One Channel**



**NOTE:** Only Channel 1 can support data trace.

## 14.2 Module Operation

### 14.2.1 General Operation

There are up to four channels in MCRC controller and for each channel there is a memory mapped PSA (Parallel Signature Analysis) Signature Register and a memory mapped CRC (Cyclic Redundancy Check) Value register. A memory can be organized into multiple sectors with each sector consisting of multiple data patterns. A data pattern can be a 8, 16, 32, or 64 bit data. MCRC module performs the signature calculation and compares the signature to a pre-determined value. The PSA Signature Register compresses an incoming data pattern into a signature when it is written. When one sector of data patterns are written into PSA Signature Register, a final signature corresponding to the sector is obtained. CRC Value Register stores the pre-determined signature corresponding to one sector of data patterns. The calculated signature and the pre-determined signature are then compared to each other for signature verification. To minimize CPU's involvement, data patterns transfer can be carried out at the background of CPU using DMA controller. DMA is setup to transfer data from memory of which the contents to be verified to the memory mapped PSA Signature Register. When DMA transfers data to the memory mapped PSA Signature Register, a signature is generated. A programmable 20-bit data pattern counter is used for each channel to define the number of data patterns to calculate for each sector. Signature verification can be performed automatically by MCRC controller in AUTO mode or by CPU itself in Semi-CPU or Full-CPU mode. In AUTO mode, a self sustained CRC signature calculation can be achieved without any CPU intervention. MCRC Controller also provides data trace capability. Channel 1 can perform data trace on CPU data bus. During data trace, channel 1 monitors any data being read on CPU data bus and compresses it. When data trace is enabled for channel 1, all circuits related to DMA request and interrupt generation and counters are disabled.

### 14.2.2 CRC Modes of Operation

MCRC Controller can operate in AUTO, Semi-CPU, Semi-CPU and Full-CPU modes.

#### 14.2.2.1 AUTO Mode

In AUTO mode, MCRC Controller in conjunction with DMA controller can perform CRC totally without CPU intervention. A sustained transfer of data to both the PSA Signature Register and CRC Value Register are performed in the background of CPU. When a mismatch is detected, an interrupt is generated to CPU. A 16 bit current sector ID register is provided to identify which sector causes a CRC failure.

#### 14.2.2.2 Semi-CPU Mode

In Semi-CPU mode, DMA controller is also utilized to perform data patterns transfer to PSA Signature Register. Instead of performing signature verification automatically, the CRC controller generates an compression complete interrupt to CPU after each sector is compressed. Upon responding to the interrupt the CPU performs the signature verification by reading the calculated signature stored at the PSA Sector Signature Register and compare it to a pre-determined CRC value.

#### 14.2.2.3 Semi-CPU Mode

In Semi-CPU mode, DMA controller is also utilized to perform data patterns transfer to PSA Signature Register. Instead of performing signature verification automatically, the CRC controller generates an compression complete interrupt to CPU after each sector is compressed. Upon responding to the interrupt the CPU performs the signature verification by reading the calculated signature stored at the PSA Sector Signature Register and compare it to a pre-determined CRC value.

#### 14.2.2.4 Full CPU Mode

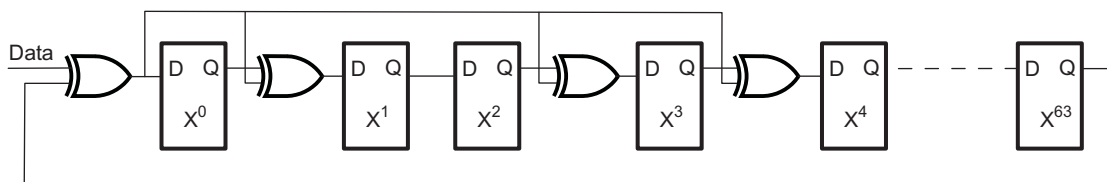
In Full-CPU mode, the CPU does the data patterns transfer and signature verification all by itself. When CPU has enough throughput, it can perform data patterns transfer by reading data from the memory system to the PSA Signature Register. After certain number of data patterns are compressed, the CPU can read from the PSA Signature Register and compare the calculated signature to the pre-determined CRC signature value. In Full-CPU mode, neither interrupt nor DMA request is generated. All counters are also disabled.

### 14.2.3 PSA Signature Register

The 64-bit PSA Signature Register is based on the primitive polynomial (as in the following equation) to produce the maximum length LFSR (Linear Feedback Shift Register).

$$f(x) = x^{64} + x^4 + x^3 + x + 1 \quad (25)$$

Figure 14-3. LFSR



The serial implementation of LFSF has a limitation that, it requires 'n' clock cycles to calculate the CRC values for an 'n' bit data stream. The idea is to produce the same CRC value operating on a multi-bit data stream, as would occur if the CRC were computed one bit at a time over the whole data stream. The algorithm involves looping to simulate the shifting, and concatenating strings to build the equations after 'n' shift.

The parallel CRC calculation based on the above polynomial can be illustrated in the below HDL code.

```

for i in 63 to 0 loop
  NEXT_CRC_VAL(0) := CRC_VAL(63) xor DATA(i);
  for j in 1 to 63 loop
    case j is
      when 1|3|4 =>
        NEXT_CRC_VAL(j) :=
          CRC_VAL(j - 1) xor CRC_VAL(63) xor DATA(i);
      when others =>
        NEXT_CRC_VAL(j) := CRC_VAL(j - 1);
    end case;
  end loop;
  CRC_VAL := NEXT_CRC_VAL;
end loop;

```

- NOTE:**
- 1) The inner loop is to calculate the next value of each shift register bit after one cycle
  - 2) The outer loop is to simulate 64 cycles of shifting. The equation for each shift register bit is thus built before it is compressed into the shift register.
  - 3) MSB of the DATA is shifted in first

There is one PSA Signature Register per CRC channel. PSA Signature Register can be both read and written. When it is written, it can either compress the data or just capture the data depending on the state of CHx\_MODE bits. If CHx\_MODE=Data Capture, a seed value can be planted in the PSA Signature Register without compression. Other modes other than Data Capture will result with the data compressed by PSA Signature Register when it is written. Each channel can be planted with different seed value before compression starts. When PSA Signature Register is read, it gives the calculated signature.

MCRC Controller should be used in conjunction with the on chip DMA controller to produce optimal system performance. The incoming data pattern to PSA Signature Register is typically initiated by the DMA master. When DMA is properly setup, it would read data from the pre-determined memory system and write them to the memory mapped PSA Signature Register. Each time PSA Signature Register is written a signature is generated. CPU itself can also perform data transfer by reading from the memory system and perform write operation to PSA Signature Register if CPU has enough throughput to handle data patterns transfer.

After system reset and when AUTO mode is enabled, MCRC Controller automatically generates a DMA request to request the pre-determined CRC value corresponding to the first sector of memory to be checked.

In AUTO mode, when one sector of data patterns is compressed, the signature stored at the PSA Signature Register is first copied to the PSA Sector Signature Register and PSA Signature Register is then cleared out to all zeros. An automatic signature verification is then performed by comparing the signature stored at the PSA Sector Signature Register to the CRC Value Register. After the comparison the MCRC Controller can generate a DMA request. Upon receiving the DMA request the DMA controller will update the CRC Value Register by transferring the next pre-determined signature value associated with the next sector of memory system. If the signature verification fails then MCRC Controller can generate a CRC fail interrupt.

In Full-CPU mode, no DMA request and interrupt are generated at all. The number of data patterns to be compressed is determined by CPU itself. Full-CPU mode is useful when DMA controller is not available to perform background data patterns transfer. The OS can periodically generate a software interrupt to CPU and use CPU to accomplish data transfer and signature verification.

MCRC Controller supports doubleword, word, half word and byte access to the PSA Signature Register. During a non-doubleword write access, all unwritten byte lanes are padded with zero's before compression. Note that comparison between PSA Sector Signature Register and CRC Value Register is always in 64 bit because a compressed value is always expressed in 64 bit.

There is a software reset per channel for PSA Signature Register. When set, the PSA Signature Register is reset to all zeros.

PSA Signature Register is reset to zero under the following conditions:

- System reset
- PSA Software reset
- One sector of data patterns are compressed

#### **14.2.4 PSA Sector Signature Register**

After one sector of data is compressed, the final resulting signature calculated by PSA Signature Register is transferred to the PSA Sector Signature Register. PSA Signature Register is a read only register. During Semi-CPU mode, the host CPU should read from the PSA Sector Signature Register instead of reading from PSA Signature Register for signature verification to avoid data coherency issue. The PSA Signature Register can be updated with new signature before the host CPU is able to retrieve it.

In Semi-CPU mode, no DMA request is generated. When one sector of data patterns is compressed, CRC controller first generates a compression complete interrupt. Responding to the interrupt, CPU will in the ISR read the PSA Sector Signature Register and compare it to the known good signature or write the signature value to another memory location to build a signature file. In Semi-CPU mode, CPU must perform the signature verification in a manner to prevent any overrun condition. The overrun condition occurs when the compression complete interrupt is generated after one sector of data patterns is compressed and CPU has not read from the PSA Sector Signature Register to perform necessary signature verification before PSA Sector Signature Register is overridden with a new value. An overrun interrupt can be enable to generate when overrun condition occurs. During Semi-CPU mode, the host CPU should read from the PSA Sector Signature Register instead of reading from PSA Signature Register for signature verification to avoid data coherency issue. The PSA Signature Register can be updated with new signature before the host CPU is able to retrieve it.

In Semi-CPU mode, no DMA request is generated. When one sector of data patterns is compressed, CRC controller first generates a compression complete interrupt. Responding to the interrupt, CPU will in the ISR read the PSA Sector Signature Register and compare it to the known good signature or write the signature value to another memory location to build a signature file. In Semi-CPU mode, CPU must perform the signature verification in a manner to prevent any overrun condition. The overrun condition occurs when the compression complete interrupt is generated after one sector of data patterns is compressed and CPU has not read from the PSA Sector Signature Register to perform necessary signature verification before PSA Sector Signature Register is overridden with a new value. An overrun interrupt can be enable to generate when overrun condition occurs.

#### **14.2.5 CRC Value Register**

Associated with each channel there is a CRC Value Register. The CRC Value Register stores the pre-determined CRC value. After one sector of data patterns is compressed by PSA Signature Register,



MCRC Controller can automatically compare the resulting signature stored at the PSA Sector Signature Register with the pre-determined value stored at the CRC Value Register if AUTO mode is enabled. If the signature verification fails, MCRC Controller can be enabled to generate a CRC fail interrupt. When the channel is set up for Semi-CPU mode, CRC controller first generates a compression complete interrupt to CPU. Upon servicing the interrupt, CPU will then read the PSA Sector Signature Register and then read the corresponding CRC value stored at another location and compare them. When the channel is set up for Semi-CPU mode, CRC controller first generates a compression complete interrupt to CPU. Upon servicing the interrupt, CPU will then read the PSA Sector Signature Register and then read the corresponding CRC value stored at another location and compare them. CPU should not read from the CRC Value Register during Semi-CPU or Full-CPU mode because the CRC Value Register is not updated during these two modes.

In AUTO mode, for first sector's signature, DMA request is generated when mode is programmed to AUTO. For subsequent sectors, DMA request is generated after each sector is compressed. Responding to the DMA request, DMA controller reloads the CRC Value Register for the next sector of memory system to be checked.

When CRC Value Register is updated with a new CRC value, an internal flag is set to indicate that CRC Value Register contains the most current value. This flag is cleared when CRC comparison is performed. Each time at the end of the final data pattern compression of a sector, MCRC Controller first checks to see if the corresponding CRC Value Register has the most current CRC value stored in it by polling the flag. If the flag is set then the CRC comparison can be performed. If the flag is not set then it means the CRC Value Register contains stale information. A CRC underrun interrupt is generated. When an underrun condition is detected, signature verification is not performed.

MCRC Controller supports doubleword, word, half word and byte access to the CRC Value Register. As noted before comparison between PSA Sector Signature Register and CRC Value Register during AUTO mode is carried out in 64 bit.

### 14.2.6 Raw Data Register

The raw or un-compressed data written to the PSA Signature Register is also saved in the Raw Data Register. This register is read only.

### 14.2.7 Example DMA Controller Setup

DMA controller needs to be setup properly in either either AUTO or Semi-CPU or Full-CPU mode as DMA controller is used to transfer data patterns. Hardware or a combination of hardware and software DMA triggering are supported.

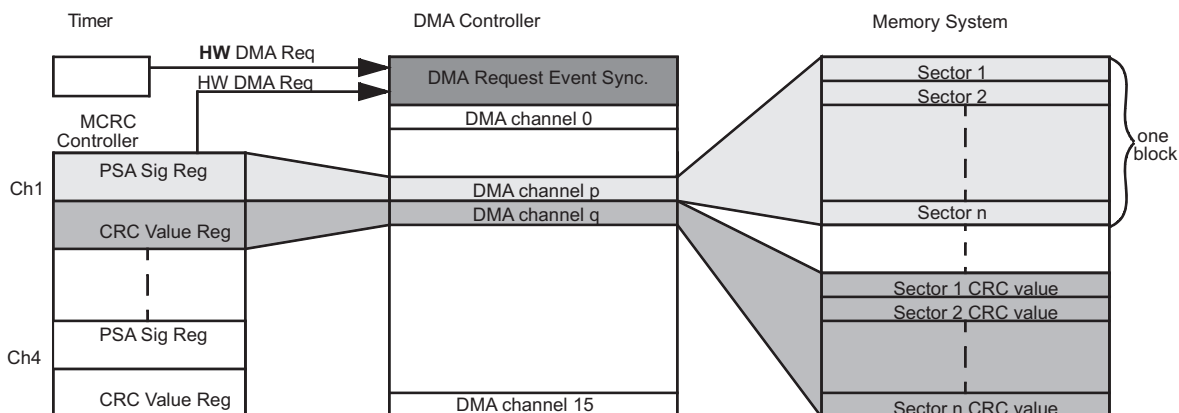
#### 14.2.7.1 AUTO Mode Using Hardware Timer Trigger

There are two DMA channels associated with each CRC channel when in AUTO mode. One DMA channel is setup to transfer data patterns from the source memory to the PSA Signature Register. The second DMA channel is setup to transfer the pre-determined signature to the CRC Value Register. The trigger source for the first DMA channel can be either by hardware or by software. As illustrated in [Figure 14-4](#) a timer can be used to trigger a DMA request to initiate transfer from the source memory system to PSA Signature Register. In AUTO mode, MCRC Controller also generates DMA request after one sector of data patterns is compressed to initiate transfer of the next CRC value corresponding to the next sector of memory. Thus a new CRC value is always updated in the CRC Value Register by DMA synchronized to each sector of memory.

A block of memory system is usually divided into many sectors. All sectors are the same size. The sector size is programmed in the CRC\_PCOUNT\_REGx and the number of sectors in one block is programmed in the CRC\_SCOUNT\_REGx of the respective channel. CRC\_PCOUNT\_REGx multiplies CRC\_SCOUNT\_REGx and multiplies transfer size of each data pattern should give the total block size in number of bytes.

The total size of the memory system to be examined is also programmed in the respective transfer count register inside DMA module. The DMA transfer count register is divided into two parts. They are element count and frame count. Note that an HW DMA request can be programmed to trigger either one frame or one entire block transfer. In Figure 14-4, an HW DMA request from a timer is used as a trigger source to initiate DMA transfer. If all four CRC channels are active in AUTO mode then a total of four DMA requests would be generated by MCRC Controller.

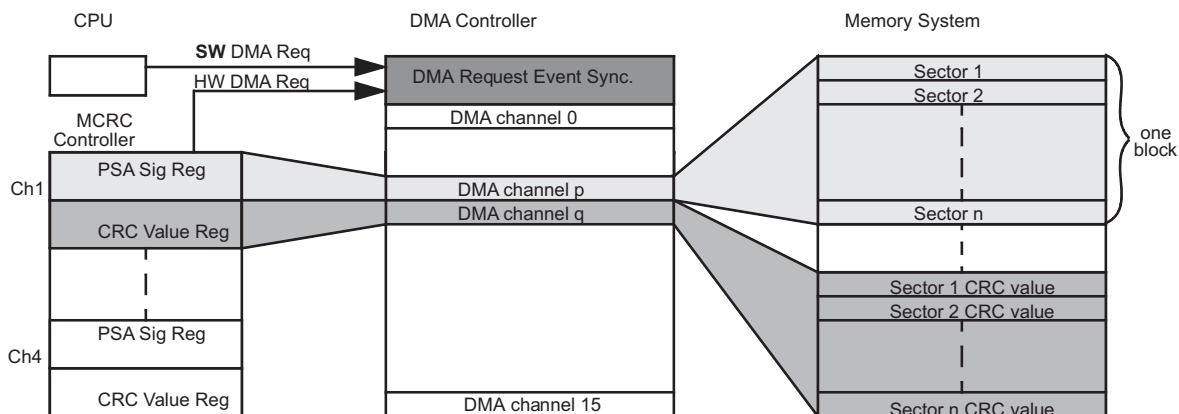
Figure 14-4. AUTO Mode Using Hardware Timer Trigger



#### 14.2.7.2 AUTO Mode Using Software Trigger

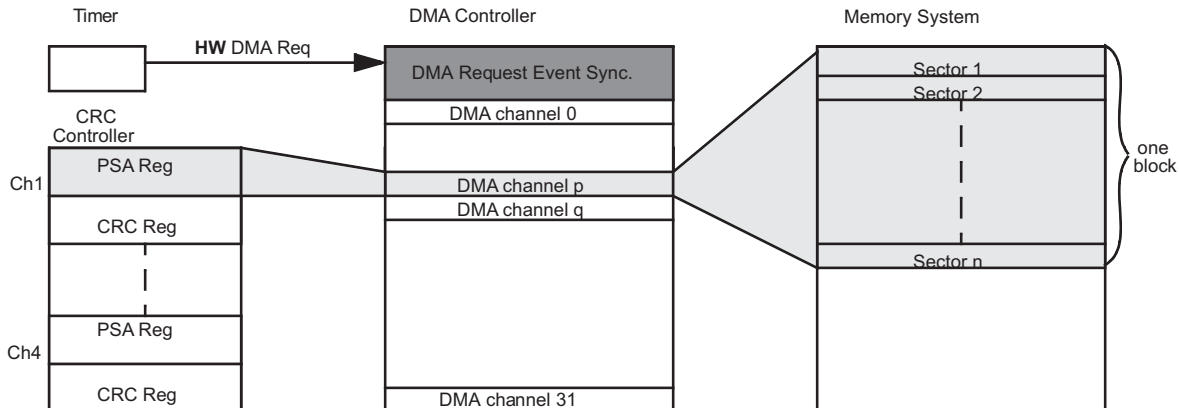
The data patterns transfer can also be initiated by software. CPU can generate a software DMA request to activate the DMA channel to transfer data patterns from source memory system to the PSA Signature Register. To generate a software DMA request CPU needs to set the corresponding DMA channel in the DMA software trigger register. Note that just one software DMA request from CPU is enough to complete the entire data patterns transfer for all sectors. Please see Figure 14-5 for illustration.

Figure 14-5. AUTO Mode With Software CPU Trigger



#### 14.2.7.3 Semi-CPU Mode Using Hardware Timer Trigger

During semi-CPU mode, no DMA request is generated by CRC controller. Therefore, no DMA channel is allocated to update CRC Value Register. CPU should not read from CRC Value Register in semi-CPU mode as it contains stale value. Note that no signature verification is performed at all during this mode. Similar to AUTO mode, either by hardware or by software DMA request can be used as a trigger for data patterns transfer. Figure 14-6 illustrates the DMA setup using semi-CPU mode with hardware timer trigger.

**Figure 14-6. Semi-CPU Mode With Hardware Timer Trigger****Table 14-1. CRC Modes in which DMA Request and Counter Logic are Active or Inactive**

	DMA request	Pattern counter	Sector counter	Timeout counter
AUTO	active	active	active	active
Semi-CPU	inactive	active	active	active
Semi-CPU	inactive	active	active	active
Full-CPU	inactive	inactive	inactive	inactive

### 14.2.8 Pattern Count Register

There is a 20-bit data pattern counter for every CRC channel. The data pattern counter is a down counter and can be pre-loaded with a programmable value stored in the Pattern Count Register. When the data pattern counter reaches zero, a compression complete interrupt is generated in Semi-CPU mode and, a compression complete interrupt is generated in Semi-CPU mode and an automatic signature verification is performed in AUTO mode. In AUTO only, DMA request is generated to trigger the DMA controller to update the CRC Value Register.

**NOTE:** The data pattern count should be divisible by the total transfer count as programmed in DMA controller. The total transfer count is the product of element count and frame count.

### 14.2.9 Sector Count Register/Current Sector Register

Each channel contains a 16 bit sector counter. The sector count register stores the number of sectors. Sector counter is a free running counter and is incremented by one each time when one sector of data patterns is compressed. When the signature verification fails, the current value stored in the sector counter is saved into current sector register. If signature verification fails, CPU can read from the current sector register to identify the sector which causes the CRC mismatch. To aid and facilitate the CPU in determining the cause of a CRC failure it is advisable to use the following equation during CRC and DMA setup.

$$\text{CRC Pattern Count} \times \text{CRC Sector Count} = \text{DMA Element Count} \times \text{DMA Frame Count}$$

The current sector register is frozen from being updated until both the current sector register is read and CRC fail status bit is cleared by CPU. If CPU does not respond to the CRC failure in a timely manner before another sector produces a signature verification failure, the current sector register is not updated with the new sector number. An overrun interrupt is generate instead. If current sector register is already frozen with an erroneous sector and emulation is entered with SUSPEND signal goes to high then the register still remains frozen even it is read.

In Semi-CPU mode, the current sector register is used to indicate the sector for which the compression complete has last happened.

The current sector register is reset when the PSA software reset is enabled.

---

**NOTE:** Both data pattern count and sector count registers must be greater than or equal to one for the counters to count. After reset, pattern count and sector count registers default to zero and the associated counters are inactive.

---

### 14.2.10 Interrupt

CRC generate several types of interrupts per channel. Associated with each interrupt, there is an interrupt enable bit. No interrupt is generated in Full-CPU mode.

- Compression complete interrupt
- CRC fail interrupt
- Overrun interrupt
- Underrun interrupt
- Timeout interrupt

**Table 14-2. Modes in which Interrupt Condition Can Occur**

	AUTO	Semi-CPU	Full-CPU
Compression Complete	no	yes	no
CRC Fail	yes	no	no
Overrun	yes	yes	no
Underrun	yes	no	no
Timeout	yes	yes	no

#### 14.2.10.1 Compression Complete Interrupt

Compression complete interrupt is generated in Semi-CPU mode only. When the data pattern counter reaches zero, the compression complete flag is set and the interrupt is generated.

#### 14.2.10.2 CRC Fail Interrupt

CRC fail interrupt is generated in AUTO mode only. When the signature verification fails, the CRC fail flag is set. CPU should take action to address the fail condition and clear the CRC fail flag after it resolves the CRC mismatch.

#### 14.2.10.3 Overrun Interrupt

Overrun interrupt is generated in either either AUTO or Semi-CPU or Semi-CPU mode. During AUTO mode, if a CRC fail is detected then the current sector number is recorded in the current sector register. If CRC fail status bit is not cleared and current sector register is not read by the host CPU before another CRC fail is detected for another sector then an overrun interrupt is generated. During Semi-CPU mode, when the data pattern counter finishes counting, it generates a compression complete interrupt. At the same time the signature is copied into the PSA Sector Signature Register. If the host CPU does not read the signature from PSA Sector Signature Register before it is updated again with a new signature value then an overrun interrupt is generated. During Semi-CPU mode, when the data pattern counter finishes counting, it generates a compression complete interrupt. At the same time the signature is copied into the PSA Sector Signature Register. If the host CPU does not read the signature from PSA Sector Signature Register before it is updated again with a new signature value then an overrun interrupt is generated.

#### 14.2.10.4 Underrun Interrupt

Underrun interrupt only occurs in AUTO mode. The interrupt is generated when the CRC Value Register is not updated with the corresponding signature when the data pattern counter finishes counting. During AUTO mode, MCRC Controller generates DMA request to update CRC Value Register in synchronization to the corresponding sector of the memory. Signature verification is also performed if underrun condition is detected. And CRC fail interrupt is generated at the same time as the underrun interrupt.

#### 14.2.10.5 Timeout Interrupt

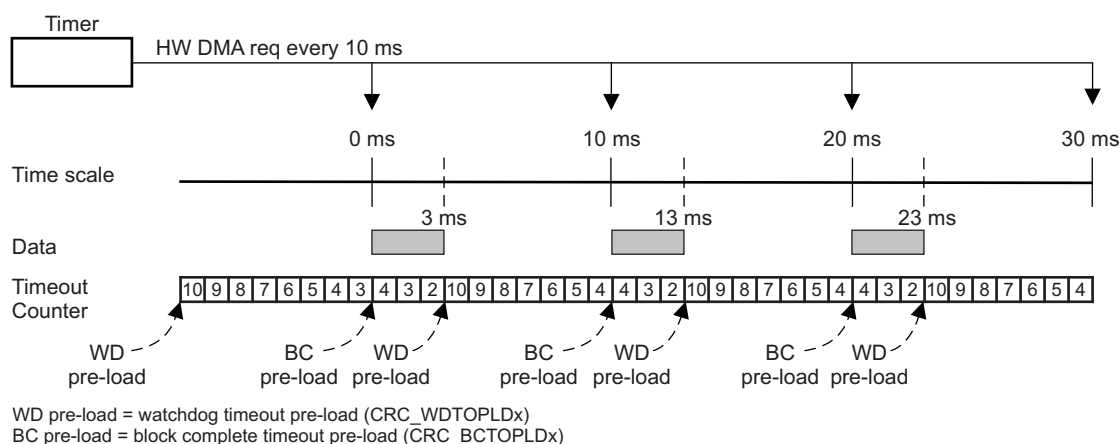
To ensure that the memory system is examined within a pre-defined time frame and no loss of incoming data there is a 24 bit timeout counter per CRC channel. The 24 bit timeout down counter can be pre-loaded with two different pre-load values, watchdog timeout pre-load value (CRC\_WDTPLDx) and block complete timeout pre-load value (CRC\_BCTOPLDx). The timeout counter is clocked by a prescaler clock which is permanently running at division 64 of HCLK clock.

Watchdog timeout pre-load register (CRC\_WDTPLDx) is used to check if DMA does supply a block of data responding to a request in a given time frame. Block complete timeout pre-load register (CRC\_BCTOPLDx) is used to check if one complete block of data patterns are compressed within a specific time frame. The timeout counter is first pre-loaded with CRC\_WDTPLDx after either either AUTO or Semi-CPU or Semi-CPU mode is selected and starts to down count. If the timeout counter expires before DMA transfers any data pattern to PSA Signature Register then a timeout interrupt is generated. An incoming data pattern before the timeout counter expires will automatically pre-load the timeout counter with CRC\_BCTOPLDx the block complete timeout pre-load value.

Block complete timeout pre-load value is used to check if one block of data patterns are compressed within a given time limit. If the timeout counter pre-loaded with CRC\_BCTOPLDx value expires before one block of data patterns are compressed a timeout interrupt is generated. When one block (pattern count x sector count) of data patterns are compressed before the counter has expired, the counter is pre-loaded with CRC\_WDTPLDx value again. If the timeout counter is pre-loaded with zero then the counter is disable and no timeout interrupt is generated.

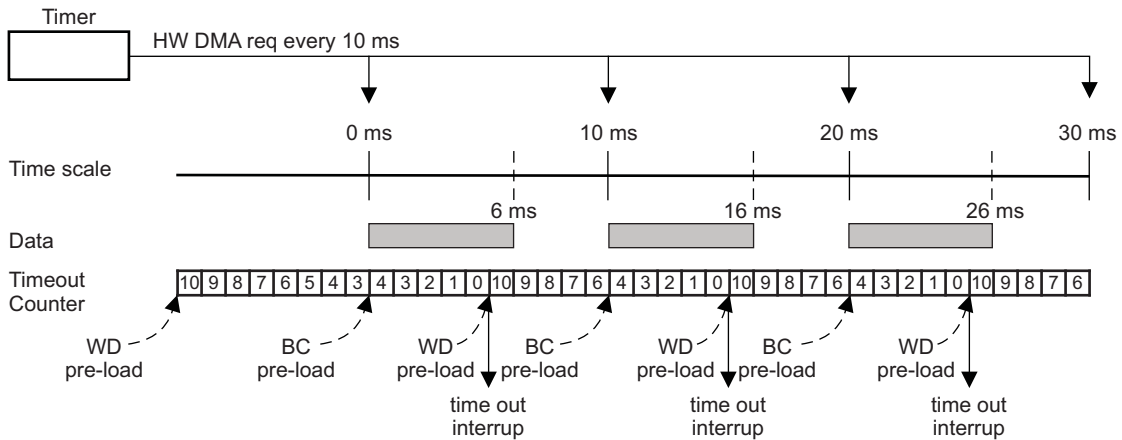
In [Figure 14-7](#), a timer generates DMA request every 10ms to trigger one block (pattern count x sector count) transfer. Since we want to make sure that DMA does start to transfer a block every 10 ms we would set the first pre-load value to 10ms in CRC\_WDTPLDx. We also want to make sure that one block of data patterns are compressed within 4ms. With such a requirement, we would set the second pre-load value to 4ms in CRC\_BCTOPLDx register.

**Figure 14-7. Timeout Example 1**



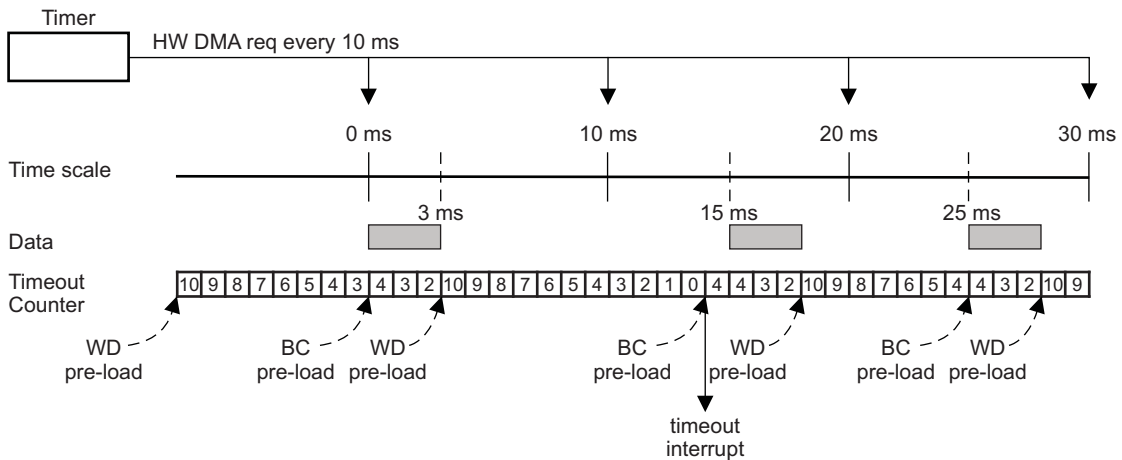
Note: No timeout interrupt is generated in this example since each block of data patterns are compressed in 3 ms and DMA does initiate a block transfer every 10 ms.

Figure 14-8. Timeout Example 2



WD pre-load = watchdog timeout pre-load (CRC\_WDTPLDx)  
 BC pre-load = block complete timeout pre-load (CRC\_BCTOPLDx)  
 Note: Timeout interrupt is generated in this example since each block of data patterns are compressed in 6 ms and this is out of the 4ms time frame.

Figure 14-9. Timeout Example 3



WD pre-load = watchdog timeout pre-load (CRC\_WDTPLDx)  
 BC pre-load = block complete timeout pre-load (CRC\_BCTOPLDx)  
 Note: Timeout interrupt is generated in this example since DMA can not transfer the second block of data within 10ms time limit and the reason may be that DMA is set up in fixed priority scheme and DMA is serving other higher priority channels at the time before it can service the timer request.

14.2.10.6 Interrupt Offset Register

MCRC Controller only generates one interrupt request to interrupt manager. A interrupt offset register is provided to indicate the source of the pending interrupt with highest priority. Table 14-3 shows the offset interrupt vector address of each interrupt condition in an ascending order of priority.

Table 14-3. Interrupt Offset Mapping

Interrupt Condition	Offset Value
Phantom	0x0
Ch1 CRC Fail	0x1
Ch2 CRC Fail	0x2
Ch3 CRC Fail	0x3
Ch4 CRC Fail	0x4

**Table 14-3. Interrupt Offset Mapping (continued)**

Interrupt Condition	Offset Value
reserved	0x5-0x8
Ch1 Compression Complete	0x9
Ch2 Compression Complete	0xa
Ch3 Compression Complete	0xb
Ch4 Compression Complete	0xc
reserved	0xd-0x10
reserved	0x5-0x8
Ch1 Compression Complete	0x9
Ch2 Compression Complete	0xa
Ch3 Compression Complete	0xb
Ch4 Compression Complete	0xc
reserved	0xd-0x10
Ch1 Overrun	0x11
Ch2 Overrun	0x12
Ch3 Overrun	0x13
Ch4 Overrun	0x14
reserved	0x15-0x18
Ch1 Underrun	0x19
Ch2 Underrun	0x1a
Ch3 Underrun	0x1b
Ch4 Underrun	0x1c
reserved	0x1d-0x20
Ch1 Timeout	0x21
Ch2 Timeout	0x22
Ch3 Timeout	0x23
Ch4 Timeout	0x24

#### 14.2.10.7 Error Handling

When an interrupt is generated, host CPU should take appropriate actions to identify the source of error and restart the respective channel in DMA and MCRC module. To restart a CRC channel, the user should perform the following steps in the ISR:

1. Write to software reset bit in *CRC\_CTRL* register to reset the respective PSA Signature Register.
2. Reset the CHx\_MODE bits to "00" in *CRC\_CTRL* register as Data capture mode.
3. Set the CHx\_MODE bits in *CRC\_CTRL* register to desired new mode again.
4. Release software reset.

The host CPU should use byte write to restart each individual channel.

#### 14.2.11 CPU Data Trace

CRC channel 1 can be used to snoop Flash, System RAM and Peripheral Bus Master Data buses. However, at any one point only one bus is snooped. It is possible to disable the snooping of any of the buses by programming MCRC\_BUS\_SEL register. While snooping the data, there is a priority scheme implemented between the buses. Peripheral Bus Master has the highest priority followed by Flash, Even System RAM and Odd System RAM. For each data read by CPU on its data bus the same data is compressed in the PSA Signature Register. A write to PSA Signature Register does not get compressed. Therefore, it is possible to write a seed value into PSA Signature Register before the bus snooping takes place. During data trace mode, all interrupts and DMA request logic are inactive. For non double word read on the data bus, all un-selected byte lanes are padded with zero during compression.



### 14.2.12 Power Down Mode

MCRC module can be put into power down mode when the power down control bit PWDN is set. The module wakes up when the PWDN bit is cleared. When MCRC controller is in power down mode, no data tracing alone will happen. However if MCRC registers are accessed then data trace happens from channel 1.

### 14.2.13 Emulation

A read access from a register in functional mode can sometimes trigger a certain internal event to follow. For example, reading an interrupt offset register triggers an event to clear the corresponding interrupt status flag. During emulation when SUSPEND signal is high, a read access from any register should only return the register contents to the bus and should not trigger or mask any event as it would have in functional mode. This is to prevent debugger from reading the interrupt offset register, i.e. during refreshing screen and cause the corresponding interrupt status flag to get cleared. Timeout counters are stopped to generate timeout interrupts in emulation mode. No Peripheral Master bus error should be generated if reading from the unimplemented locations. When channel 1 is placed under data trace, the PSA Signature Register does not compress any data read on CPU data bus when suspend is active.

### 14.2.14 Peripheral Bus Interface

MCRC is a Peripheral slave module. The register interface is similar to other peripheral modules. MCRC supports following features:

- Different sizes of burst operation.
- Aligned and unaligned accesses.
- Returns "Address Error" if the address accessed is beyond specific region. Refer Register table for valid address region.
- CEMUDEBUG signal is used to put MCRC in suspend mode.



### 14.3 Example

This section illustrates several of the ways in which the MCRC Controller can be utilized to perform CRC.

#### 14.3.1 Example: Auto Mode Using Time Based Event Triggering

A large memory area with 2Mbyte (256k doubleword) is to be checked in the background of CPU. CRC is to be performed every 1K byte (128 doubleword). Therefore there should be 2048 pre-recorded CRC values. For illustration purpose, we map channel 1 CRC Value Register to DMA channel 1 and channel 1 PSA Signature Register to DMA channel 2. Assume all DMA transfers are carried out in 64-bit transfer size.

##### 14.3.1.1 DMA Setup

- Set up DMA channel 1 with the starting address from which the pre-determined CRC values are stored. Set up the destination address to the memory mapped channel 1 CRC Value Register. Put the source address at post increment addressing mode and put the destination address at constant addressing mode. Use **hardware** DMA request for channel 1 to trigger a **frame** transfer.
- Set up DMA channel 2 with the source address from which the contents of memory to be verified. Set up the destination address to the memory mapped channel 1 PSA Signature Register. Program the element transfer count to 128 and the frame transfer count to 2048. Put the source address at post increment addressing mode and put the destination address at constant address mode. Use **hardware** DMA request for channel 2 to trigger an entire **block** transfer.

##### 14.3.1.2 Timer Setup

The timer can be any general purpose timer which is capable of generating a time based DMA request.

- Set up Timer to generate DMA request associated with DMA channel 2. For example, an OS can setup the timer to generate a DMA request every 10ms.

##### 14.3.1.3 CRC Setup

- Program the pattern count to 128.
- Program the sector count to 2048.
- For example, we want the entire 2Mbytes to be compressed within 5ms. We can program the block complete timeout pre-load (CRC\_BCTOPLDx) value to 15625 (5 ms / (1 HCLK period x 64)) if CRC is operating at 200 Mhz.
- Enable AUTO mode and all interrupts.

After AUTO mode is selected, MCRC Controller automatically generates a DMA request on channel 1. Around the same time the timer module also generates a DMA request on DMA channel 2. When the first incoming data pattern arrives at the PSA Signature Register, the MCRC Controller will compress it. After some time, the DMA controller would update the CRC Value Register with a pre-determined value matching the calculated signature for the first sector of 128 64 bit data patterns. After one sector of data patterns are compressed, the MCRC Controller generate a CRC fail interrupt if signature stored at the PSA Sector Signature Register does not match the CRC Value Register. MCRC Controller generates a DMA request on DMA channel 1 when one sector of data patterns are compressed. This routine will continue until the entire 2Mbyte are consumed. If the timeout counter reached zero before the entire 2Mbytes are compressed a timeout interrupt is generated. After 2M are transferred, the DMA can generate an interrupt to CPU. The entire operation will continue again when DMA responds to the DMA request from both the timer and MCRC Controller. The CRC is performed totally without any CPU intervention.

#### 14.3.2 Example: Auto Mode Without Using Time Based Triggering

A small but highly secured memory area with 1kbytes is to be checked in the background of CPU. CRC is to be performed every 1Kbytes. Therefore there is only one pre-recorded CRC value. For illustration purpose, we map channel 1 CRC Value Register to DMA channel 1 and channel 1 PSA Signature Register to DMA channel 2. Assume all transfers carried out by DMA are in 64 bit transfer size.

### 14.3.2.1 DMA Setup

- Set up DMA channel 1 with the source address from which the pre-determined CRC value is stored. Set up the destination address to the memory mapped channel 1 CRC Value Register. Put the source address at constant addressing mode and put the destination address at constant addressing mode. Use **hardware** DMA request for channel 1.
- Set up DMA channel 2 with the source address from which the memory area to be verified. Set up the destination address to the memory mapped channel 1 PSA Signature Register. Program the element transfer count to 128 and the frame transfer count to 1. Put the source address at post increment addressing mode and put the destination address at constant address mode. Generate a **software** DMA request on channel 2 after CRC has completed its setup. Enable autoinitiation for DMA channel 2.

### 14.3.2.2 CRC Setup

- Program the pattern count to 128.
- Program the sector count to 1.
- Leaving the timeout count register with the reset value of zero means no timeout interrupt is generated.
- Enable AUTO mode and all interrupts.

After AUTO mode is selected, the MCRC Controller automatically generates a DMA request on channel 1. At the same time the CPU generates a **software** DMA request on DMA channel 2. When the first incoming data pattern arrives at the PSA Signature Register, the MCRC Controller will compress it. After some time, the DMA controller would update the CRC Value Register with a pre-determined value matching the calculated signature for the first sector of 128 64 bit data patterns. After one sector of data patterns are compressed, the MCRC Controller generate a CRC fail interrupt if signature stored at the PSA Sector Signature Register does not match the CRC Value Register. MCRC Controller generate a DMA request on DMA channel 1 again after one sector is compressed. After 1kbytes are transferred, the DMA can generate an interrupt to CPU. Responding to the DMA interrupt CPU can restart the CRC routine by generating a software DMA request onto channel 2 again.

### 14.3.3 Example: Semi-CPU Mode

If DMA controller is available in a system, the CRC module can also operate in semi-CPU mode. This means that CPU can still make use of the DMA to perform data patterns transfer to CRC controller in the background. The difference between semi-CPU mode and AUTO mode is that CRC controller does not automatically perform the signature verification. CRC controllers generates a compression complete interrupt to CPU when the one sector of data patterns are compressed. CPU needs to perform the signature verification itself.

A memory area with 2Mbyte is to be verified with the help of the CPU. CRC operation is to be performed every 1K byte. Since there are 2Mbyte (256k doublewords) of memory to be check and we want to perform a CRC every 1Kbyte (128 doublewords) and therefore there should be 2048 pre-recorded CRC values. In Semi-CPU mode, the CRC Value Register is not updated and contains indeterminate data.

#### 14.3.3.1 DMA Setup

Set up DMA channel 1 with the source address from which the memory area to be verified are mapped. Set up the destination address to the memory mapped channel 1 PSA Signature Register. Put the starting address at post increment addressing mode and put the destination address at constant address mode. Use hardware DMA request to trigger an entire block transfer for channel 1. Disable autoinitiation for DMA channel 1.

#### 14.3.3.2 Timer Setup

The timer can be any general purpose timer which is capable of generating a time based DMA request.

Setup Timer to generate DMA request associated with DMA channel 1. For example, an OS can setup the timer to generate a DMA request every 10ms.

### 14.3.3.3 CRC Setup

- Program the pattern count to 128.
- Program the sector count to 2048.
- For example, we want the entire 2Mbytes to be compressed within 5ms. We can program the block complete timeout pre-load value to 15625 (5 ms / (1 HCLK period x 64)) if CRC is operating at 200 Mhz.
- Enable Semi-CPU mode and enable all interrupts.

The timer module first generates a DMA request on DMA channel 1 when it is enabled. When the first incoming data pattern arrives at the PSA Signature Register, the CRC controller will compress it. After one sector of data patterns are compressed, the CRC controller generate a compression complete interrupt. Upon responding to the interrupt the CPU would read from the PSA Sector Signature Register. It is up to the CPU on how to deal with the PSA value just read. It can compare it to a known signature value or it can write it to another memory location to build a signature file or even transfer the signature out of the device via SCI or SPI. This routine will continue until the entire 2Mbyte are consumed. The latency of the interrupt response from CPU can cause overrun condition. If CPU does not read from PSA Sector Signature Register before the PSA value is overridden with the signature of the next sector of memory, an overrun interrupt will be generated by CRC controller.

### 14.3.4 Example: Full-CPU Mode

In a system without the availability of DMA controller, the CRC routine can be operated by CPU provided the CPU has enough throughput. CPU needs to read from the memory area from which CRC is to be performed.

A memory area with 2Mbyte is to be checked with the help of the CPU. CRC operation is to be performed every 1K byte. In CPU mode, the CRC Value Register is not updated and contains indeterminate data.

#### 14.3.4.1 CRC Setup

- All control registers can be left in their reset state. Only enable Full-CPU mode.

CPU itself reads from the memory and write the data to the PSA Signature Register inside MCRC Controller. When the first incoming data pattern arrives at the PSA Signature Register, the MCRC Controller will compress it. After  $n$  data patterns are compressed, CPU can read from the PSA Signature Register. It is up to the CPU on how to deal with the PSA signature value just read. It can compare it to a known signature value stored at another memory location.

## 14.4 CRC Control Registers

All registers are in word boundary. 64, 32, 16, and 8 bit write accesses are supported to all registers. During double word, write CADDRESS[2:0] are ignored from decoding. A double word write access can write to two 32-bit words at the same time.

Write access takes one Peripheral clock and read access takes two Peripheral clock.

**Table 14-4. CRC Control Registers**

Offset	Acronym	Register Description	Section
00	CRC_CTRL0	CRC Global Control Register	<a href="#">Section 14.4.1</a>
08h	CRC_CTRL1	CRC Global Control Register 1	<a href="#">Section 14.4.2</a>
10h	CRC_CTRL2	CRC Global Control Register 2	<a href="#">Section 14.4.3</a>
18h	CRC_INTS	CRC Interrupt Enable Set Register	<a href="#">Section 14.4.4</a>
20h	CRC_INTR	CRC Interrupt Enable Reset Register	<a href="#">Section 14.4.5</a>
28h	CRC_STATUS	CRC Interrupt Status Register	<a href="#">Section 14.4.6</a>
30h	CRC_INT_OFFSETS_ET_REG	CRC Interrupt Offset	<a href="#">Section 14.4.7</a>
38h	CRC_BUSY	CRC Busy Register	<a href="#">Section 14.4.8</a>
40h	CRC_PCOUNT_REG1	CRC Pattern Counter Preload Register	<a href="#">Section 14.4.9</a>
44h	CRC_SCOUNT_REG1	CRC Sector Counter Preload Register	<a href="#">Section 14.4.10</a>
48h	CRC_CURSEC_REG1	CRC Current Sector Register 1	<a href="#">Section 14.4.11</a>
4Ch	CRC_WDTPD1	CRC Channel 1 Watchdog Timeout Preload Register	<a href="#">Section 14.4.12</a>
50h	CRC_BCTOPD1	CRC Channel 1 Block Complete Timeout Preload Register	<a href="#">Section 14.4.13</a>
60h	PSA_SIGREGL1	Channel 1 PSA Signature Low Register	<a href="#">Section 14.4.14</a>
64h	PSA_SIGREGH1	Channel 1 PSA Signature High Register	<a href="#">Section 14.4.15</a>
68h	CRC_REGL1	Channel 1 CRC Value Low Register	<a href="#">Section 14.4.16</a>
6Ch	CRC_REGH1	Channel 1 CRC Value High Register	<a href="#">Section 14.4.17</a>
70h	PSA_SECSIGREGL1	Channel 1 PSA Sector Signature Low Register	<a href="#">Section 14.4.18</a>
74h	PSA_SECSIGREGH1	Channel 1 PSA Sector Signature High Register	<a href="#">Section 14.4.19</a>
78h	RAW_DATAREGL1	Channel 1 Raw Data Low Register	<a href="#">Section 14.4.20</a>
7Ch	RAW_DATAREGH1	Channel 1 Raw Data High Register	<a href="#">Section 14.4.21</a>
80h	CRC_PCOUNT_REG2	CRC Pattern Counter Preload Register 2	<a href="#">Section 14.4.22</a>
84h	CRC_SCOUNT_REG2	CRC Sector Counter Preload Register 2	<a href="#">Section 14.4.23</a>
88h	CRC_CURSEC_REG2	CRC Current Sector Register 2	<a href="#">Section 14.4.24</a>
8Ch	CRC_WDTPD2	CRC Channel 2 Watchdog Timeout Preload Register A	<a href="#">Section 14.4.25</a>
90h	CRC_BCTOPD2	CRC Channel 2 Block Complete Timeout Preload Register B	<a href="#">Section 14.4.26</a>
A0h	PSA_SIGREGL2	Channel 2 PSA Signature Low Register	<a href="#">Section 14.4.27</a>
A4h	PSA_SIGREGH2	Channel 2 PSA Signature High Register	<a href="#">Section 14.4.28</a>
A8h	CRC_REGL2	Channel 2 CRC Value Low Register	<a href="#">Section 14.4.29</a>
ACh	CRC_REGH2	Channel 2 CRC Value High Register	<a href="#">Section 14.4.30</a>
B0h	PSA_SECSIGREGL2	Channel 2 PSA Sector Signature Low Register	<a href="#">Section 14.4.31</a>
B4h	PSA_SECSIGREGH2	Channel 2 PSA Sector Signature High Register	<a href="#">Section 14.4.32</a>
B8h	RAW_DATAREGL2	Channel 2 Raw Data Low Register	<a href="#">Section 14.4.33</a>
BCh	RAW_DATAREGH2	Channel 2 Raw Data High Register	<a href="#">Section 14.4.34</a>
C0h	CRC_PCOUNT_REG3	CRC Pattern Counter Preload Register 3	<a href="#">Section 14.4.35</a>
C4h	CRC_SCOUNT_REG3	CRC Sector Counter Preload Register 3	<a href="#">Section 14.4.36</a>
C8h	CRC_CURSEC_REG3	CRC Current Sector Register 3	<a href="#">Section 14.4.37</a>
CCh	CRC_WDTPD3	CRC Channel 3 Watchdog Timeout Preload Register A	<a href="#">Section 14.4.38</a>
D0h	CRC_BCTOPD3	CRC Channel 4 Block Complete Timeout Preload Register B	<a href="#">Section 14.4.39</a>
E0h	PSA_SIGREGL3	Channel 3 PSA Signature Low Register	<a href="#">Section 14.4.40</a>
E4h	PSA_SIGREGH3	Channel 3 PSA Signature High Register	<a href="#">Section 14.4.41</a>
E8h	CRC_REGL3	Channel 3 CRC Value Low Register	<a href="#">Section 14.4.42</a>

**Table 14-4. CRC Control Registers (continued)**

Offset	Acronym	Register Description	Section
ECh	CRC_REGH3	Channel 3 CRC Value High Register	<a href="#">Section 14.4.43</a>
F0h	PSA_SECSIGREGL3	Channel 3 PSA Sector Signature Low Register	<a href="#">Section 14.4.44</a>
F4h	PSA_SECSIGREGH3	Channel 3 PSA Sector Signature High Register	<a href="#">Section 14.4.45</a>
F8h	RAW_DATAREGL3	Channel 3 Raw Data Low Register	<a href="#">Section 14.4.46</a>
FCh	RAW_DATAAREGH3	Channel 3 Raw Data High Register	<a href="#">Section 14.4.47</a>
100h	CRC_PCOUNT_REG4	CRC Pattern Counter Preload Register 4	<a href="#">Section 14.4.48</a>
104h	CRC_SCOUNT_REG4	CRC Sector Counter Preload Register 4	<a href="#">Section 14.4.49</a>
108h	CRC_CURSEC_REG4	CRC Current Sector Register 4	<a href="#">Section 14.4.50</a>
10Ch	CRC_WDTPD4	CRC Channel 4 Watchdog Timeout Preload Register A	<a href="#">Section 14.4.51</a>
110h	CRC_BCTOPD4	CRC Channel 4 Block Complete Timeout Preload Register B	<a href="#">Section 14.4.52</a>
120h	PSA_SIGREGL4	Channel 4 PSA Signature Low Register	<a href="#">Section 14.4.53</a>
124h	PSA_SIGREGH4	Channel 4 PSA Signature High Register	<a href="#">Section 14.4.54</a>
128h	CRC_REGL4	Channel 4 CRC Value Low Register	<a href="#">Section 14.4.55</a>
12Ch	CRC_REGH4	Channel 4 CRC Value High Register	<a href="#">Section 14.4.56</a>
130h	PSA_SECSIGREGL4	Channel 4 PSA Sector Signature Low Register	<a href="#">Section 14.4.57</a>
134h	PSA_SECSIGREGH4	Channel 4 PSA Sector Signature High Register	<a href="#">Section 14.4.58</a>
138h	RAW_DATAREGL4	Channel 4 Raw Data Low Register	<a href="#">Section 14.4.59</a>
13Ch	RAW_DATAAREGH4	Channel 4 Raw Data High Register	<a href="#">Section 14.4.60</a>
140h	MCRC_BUS_SEL	Data Bus Selection Register	<a href="#">Section 14.4.61</a>

**14.4.1 CRC Global Control Register 0 (CRC\_CTRL0)**
**Figure 14-10. CRC Global Control Register 0 (CRC\_CTRL0) [offset = 00]**

31	25	24
Reserved	R-0	CH4_PSA_SWREST
		R/W-0
23	17	16
Reserved	R-0	CH3_PSA_SWREST
		R/W-0
15	9	8
Reserved	R-0	CH2_PSA_SWREST
		R/W-0
7	1	0
Reserved	R-0	CH1_PSA_SWREST
		R/W-0

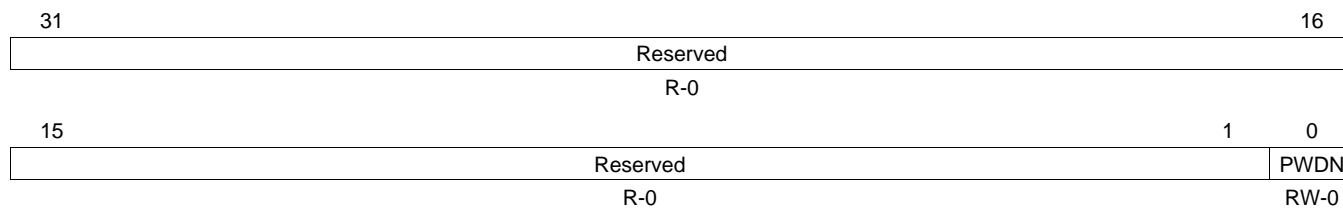
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-5. CRC Global Control Register 0 (CRC\_CTRL0) Field Descriptions**

Bit	Field	Value	Description
31-25	Reserved	0	Read returns 0. Writes have no effect.
24	CH4_PSA_SWREST	0	<b>Channel 4 PSA Software Reset.</b> When set, the PSA Signature Register is reset to all zero. Software reset does not reset software reset bit itself. Therefore, CPU is required to clear this bit by writing a '0'
		1	PSA Signature Register reset
23-17	Reserved	0	Read returns 0. Writes have no effect.
24	CH3_PSA_SWREST	0	<b>Channel 3 PSA Software Reset.</b> When set, the PSA Signature Register is reset to all zero. Software reset does not reset software reset bit itself. Therefore, CPU is required to clear this bit by writing a '0'
		1	PSA Signature Register reset
15-9	Reserved	0	Read returns 0. Writes have no effect.
8	CH2_PSA_SWREST	0	<b>Channel 2 PSA Software Reset.</b> When set, the PSA Signature Register is reset to all zero. Software reset does not reset software reset bit itself. Therefore, CPU is required to clear this bit by writing a '0'
		1	PSA Signature Register reset
7-1	Reserved	0	Read returns 0. Writes have no effect.
0	CH1_PSA_SWREST	0	<b>Channel 1 PSA Software Reset.</b> When set, the PSA Signature Register is reset to all zero. Software reset does not reset software reset bit itself. Therefore, CPU is required to clear this bit by writing a '0'
		1	PSA Signature Register reset

### 14.4.2 CRC Global Control Register (CRC\_CTRL1)

Figure 14-11. CRC Global Control Register 1 (CRC\_CTRL1) [offset = 08h]



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

Table 14-6. CRC Global Control Register 1 (CRC\_CTRL1) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved		Read returns 0. Writes have no effect.
0	PWDN	0	<b>Power Down.</b> When set, MCRC moduleMCRC Module is put in power down mode
		1	MCRC is not in power down mode
			MCRC is in power down mode

### 14.4.3 CRC Global Control Register 2 (CRC\_CTRL2)

**Figure 14-12. CRC Global Control Register 2 (CRC\_CTRL2) [offset = 10h]**

31	Reserved	26	25	24
	R-0			R/W-0
23	Reserved	18	17	16
	R-0			R/W-0
15	Reserved	10	9	8
	R-0			R/W-0
7	Reserved	5	4	3
	R-0		CH1_TRACEE N	Reserved
			R/W-0	R-0
				2
				1
				0
				R/W-0
				CH1_MODE
				R/W-0

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

**Table 14-7. CRC Global Control Register 2 (CRC\_CTRL2) Field Descriptions**

Bit	Field	Value	Description
31-26	Reserved	0	Read returns 0. Writes have no effect.
25-24	CH4_MODE	0	Channel 4 Mode Selection Data Capture mode. In this mode, the PSA Signature Register does not compress data when it is written. Any data written to PSA Signature Register is simply captured by PSA Signature Register without any compression. This mode can be used to plant seed value into the PSA register.
		1h	AUTO Mode
		2h	Semi-CPU Mode
		3h	Full-CPU Mode
23-18	Reserved	0	Read returns 0. Writes have no effect.
17-16	CH3_MODE	0	Channel 3 Mode Selection Data Capture mode. In this mode, the PSA Signature Register does not compress data when it is written. Any data written to PSA Signature Register is simply captured by PSA Signature Register without any compression. This mode can be used to plant seed value into the PSA register.
		1h	AUTO Mode
		2h	Semi-CPU Mode
		3h	Full-CPU Mode
15-10	Reserved	0	Read returns 0. Writes have no effect.
9-8	CH2_MODE	0	Channel 2 Mode Selection Data Capture mode. In this mode, the PSA Signature Register does not compress data when it is written. Any data written to PSA Signature Register is simply captured by PSA Signature Register without any compression. This mode can be used to plant seed value into the PSA register.
		1h	AUTO Mode
		2h	Semi-CPU Mode
		3h	Full-CPU Mode
7-5	Reserved		Read returns 0. Writes have no effect.



**Table 14-7. CRC Global Control Register 2 (CRC\_CTRL2) Field Descriptions (continued)**

Bit	Field	Value	Description
4	CH1_TRACEEN	0 1	<b>Channel 1 Data Trace Enable.</b> When set, the channel is put into data trace mode. The channel snoops on the CPU Peripheral Bus Master, Flash, System RAM buses for any read transaction. Any read data on these buses is compressed by the PSA Signature Register. When suspend is on, the PSA Signature Register does not compress any read data on these buses. Data Trace disable Data Trace enable
3-2	Reserved	0	Read returns 0. Writes have no effect.
1-0	CH1_MODE	0 1h 2h 3h	Channel 1 Mode Selection Data Capture mode. In this mode, the PSA Signature Register does not compress data when it is written. Any data written to PSA Signature Register is simply captured by PSA Signature Register without any compression. This mode can be used to plant seed value into the PSA register. AUTO Mode Semi-CPU Mode Full-CPU Mode

### 14.4.4 CRC Interrupt Enable Set Register (CRC\_INTS)

**Figure 14-13. CRC Interrupt Enable Set Register (CRC\_INTS) [offset = 18h]**

31	29	28	27	26	25	24
Reserved		CH4_TIMEOUT ENS	CH4_UNDERE NS	CH4_OVEREN S	CH4_CRCFAIL ENS	CH4_CCITENS
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	21	20	19	18	17	16
Reserved		CH3_TIMEOUT ENS	CH3_UNDERE NS	CH3_OVEREN S	CH3_CRCFAIL ENS	CH3_CCITENS
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	13	12	11	10	9	8
Reserved		CH2_TIMEOUT ENS	CH2_UNDERE NS	CH2_OVEREN S	CH2_CRCFAIL ENS	CH2_CCITENS
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	5	4	3	2	1	0
Reserved		CH1_TIMEOUT ENS	CH1_UNDERE NS	CH1_OVEREN S	CH1_CRCFAIL ENS	CH1_CCITENS
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

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

**Table 14-8. CRC Interrupt Enable Set Register (CRC\_INTS) Field Descriptions**

Bit	Field	Value	Description
31-29	Reserved	0	Read returns 0. Writes have no effect.
28	CH4_TIMEOUTENS	0 1	<p><b>Channel 4 Timeout Interrupt Enable Bit.</b> Writing a one to this bit enable the timeout interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).</p> <p>User and Privileged mode read</p> <p>0 Timeout Interrupt disable 1 Timeout Interrupt enable</p> <p>User and Privileged mode write</p> <p>0 Has no effect 1 Timeout Interrupt enable</p>
27	CH4_UNDERENS	0 1	<p><b>Channel 4 Underrun Interrupt Enable Bit.</b> Writing a one to this bit enable the underrun interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).</p> <p>User and Privileged mode read</p> <p>0 Underrun Interrupt disable 1 Underrun Interrupt enable</p> <p>User and Privileged mode write</p> <p>0 Has no effect 1 Underrun Interrupt enable</p>
26	CH4_OVERENS	0 1	<p><b>Channel 4 Overrun Interrupt Enable Bit.</b> Writing a one to this bit enable the overrun interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).</p> <p>User and Privileged mode read</p> <p>0 Overrun Interrupt disable 1 Overrun Interrupt enable</p> <p>User and Privileged mode write</p> <p>0 Has no effect 1 Overrun Interrupt enable</p>

**Table 14-8. CRC Interrupt Enable Set Register (CRC\_INTS) Field Descriptions (continued)**

Bit	Field	Value	Description
25	CH4_CRCFAILSENS	<p>User and Privileged mode read</p> <p>0 CRC Fail Interrupt disable</p> <p>1 CRC Fail Interrupt enable</p> <p>User and Privileged mode write</p> <p>0 Has no effect</p> <p>1 CRC Fail Interrupt enable</p>	<p><b>Channel 4 CRC Fail Interrupt Enable Bit.</b> Writing a one to this bit enable the CRC fail interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).</p>
24	CH4_CCITENS	<p>User and Privileged mode read</p> <p>0 Compression Complete Interrupt disable</p> <p>1 Compression Complete Interrupt enable</p> <p>User and Privileged mode write</p> <p>0 Has no effect</p> <p>1 Compression Complete Interrupt enable</p>	<p><b>Channel 4 Compression Complete Interrupt Enable Bit.</b> Writing a one to this bit enable the CRC fail interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).</p>
23-21	Reserved		Read returns 0. Writes have no effect.
20	CH3_TIMEOUTSENS	<p>User and Privileged mode read</p> <p>0 Timeout Interrupt disable</p> <p>1 Timeout Interrupt enable</p> <p>User and Privileged mode write</p> <p>0 Has no effect</p> <p>1 Timeout Interrupt enable</p>	<p><b>Channel 3 Timeout Interrupt Enable Bit.</b> Writing a one to this bit enable the timeout interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).</p>
19	CH3_UNDERSENS	<p>User and Privileged mode read</p> <p>0 Underrun Interrupt disable</p> <p>1 Underrun Interrupt enable</p> <p>User and Privileged mode write</p> <p>0 Has no effect</p> <p>1 Underrun Interrupt enable</p>	<p><b>Channel 3 Underrun Interrupt Enable Bit.</b> Writing a one to this bit enable the underrun interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).</p>
18	CH3_OVERSENS	<p>User and Privileged mode read</p> <p>0 Overrun Interrupt disable</p> <p>1 Overrun Interrupt enable</p> <p>User and Privileged mode write</p> <p>0 Has no effect</p> <p>1 Overrun Interrupt enable</p>	<p><b>Channel 3 Overrun Interrupt Enable Bit.</b> Writing a one to this bit enable the overrun interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).</p>

**Table 14-8. CRC Interrupt Enable Set Register (CRC\_INTS) Field Descriptions (continued)**

Bit	Field	Value	Description
17	CH3_CRCFAILSENS	<p>User and Privileged mode read</p> <p>0 CRC Fail Interrupt disable</p> <p>1 CRC Fail Interrupt enable</p> <p>User and Privileged mode write</p> <p>0 Has no effect</p> <p>1 CRC Fail Interrupt enable</p>	<p><b>Channel 3 CRC Fail Interrupt Enable Bit.</b> Writing a one to this bit enable the CRC fail interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).</p>
16	CH3_CCITENS	<p>User and Privileged mode read</p> <p>0 Compression Complete Interrupt disable</p> <p>1 Compression Complete Interrupt enable</p> <p>User and Privileged mode write</p> <p>0 Has no effect</p> <p>1 Compression Complete Interrupt enable</p>	<p><b>Channel 3 Compression Complete Interrupt Enable Bit.</b> Writing a one to this bit enable the CRC fail interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).</p>
15-13	Reserved		Read returns 0. Writes have no effect.
12	CH2_TIMEOUTSENS	<p>User and Privileged mode read</p> <p>0 Timeout Interrupt disable</p> <p>1 Timeout Interrupt enable</p> <p>User and Privileged mode write</p> <p>0 Has no effect</p> <p>1 Timeout Interrupt enable</p>	<p><b>Channel 2 Timeout Interrupt Enable Bit.</b> Writing a one to this bit enable the timeout interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).</p>
11	CH2_UNDERSENS	<p>User and Privileged mode read</p> <p>0 Underrun Interrupt disable</p> <p>1 Underrun Interrupt enable</p> <p>User and Privileged mode write</p> <p>0 Has no effect</p> <p>1 Underrun Interrupt enable</p>	<p><b>Channel 2 Underrun Interrupt Enable Bit.</b> Writing a one to this bit enable the underrun interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).</p>
10	CH2_OVERSENS	<p>User and Privileged mode read</p> <p>0 Overrun Interrupt disable</p> <p>1 Overrun Interrupt enable</p> <p>User and Privileged mode write</p> <p>0 Has no effect</p> <p>1 Overrun Interrupt enable</p>	<p><b>Channel 2 Overrun Interrupt Enable Bit.</b> Writing a one to this bit enable the overrun interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).</p>

**Table 14-8. CRC Interrupt Enable Set Register (CRC\_INTS) Field Descriptions (continued)**

Bit	Field	Value	Description
9	CH2_CRCFAILSENS	<p>User and Privileged mode read</p> <p>0 CRC Fail Interrupt disable</p> <p>1 CRC Fail Interrupt enable</p> <p>User and Privileged mode write</p> <p>0 Has no effect</p> <p>1 CRC Fail Interrupt enable</p>	<p><b>Channel 2 CRC Fail Interrupt Enable Bit.</b> Writing a one to this bit enable the CRC fail interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).</p>
8	CH2_CCITENS	<p>User and Privileged mode read</p> <p>0 Compression Complete Interrupt disable</p> <p>1 Compression Complete Interrupt enable</p> <p>User and Privileged mode write</p> <p>0 Has no effect</p> <p>1 Compression Complete Interrupt enable</p>	<p><b>Channel 2 Compression Complete Interrupt Enable Bit.</b> Writing a one to this bit enable the CRC fail interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).</p>
7-5	Reserved		Read returns 0. Writes have no effect.
4	CH1_TIMEOUTSENS	<p>User and Privileged mode read</p> <p>0 Timeout Interrupt disable</p> <p>1 Timeout Interrupt enable</p> <p>User and Privileged mode write</p> <p>0 Has no effect</p> <p>1 Timeout Interrupt enable</p>	<p><b>Channel 1 Timeout Interrupt Enable Bit.</b> Writing a one to this bit enable the timeout interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).</p>
3	CH1_UNDERSENS	<p>User and Privileged mode read</p> <p>0 Underrun Interrupt disable</p> <p>1 Underrun Interrupt enable</p> <p>User and Privileged mode write</p> <p>0 Has no effect</p> <p>1 Underrun Interrupt enable</p>	<p><b>Channel 1 Underrun Interrupt Enable Bit.</b> Writing a one to this bit enable the underrun interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).</p>
2	CH1_OVERSENS	<p>User and Privileged mode read</p> <p>0 Overrun Interrupt disable</p> <p>1 Overrun Interrupt enable</p> <p>User and Privileged mode write</p> <p>0 Has no effect</p> <p>1 Overrun Interrupt enable</p>	<p><b>Channel 1 Overrun Interrupt Enable Bit.</b> Writing a one to this bit enable the overrun interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).</p>

**Table 14-8. CRC Interrupt Enable Set Register (CRC\_INTS) Field Descriptions (continued)**

Bit	Field	Value	Description
1	CH1_CRCFAILENS		<p><b>Channel 1 CRC Fail Interrupt Enable Bit.</b> Writing a one to this bit enable the CRC fail interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).</p> <p>User and Privileged mode read</p> <p>0 CRC Fail Interrupt disable</p> <p>1 CRC Fail Interrupt enable</p> <p>User and Privileged mode write</p> <p>0 Has no effect</p> <p>1 CRC Fail Interrupt enable</p>
0	CH1_CCITENS		<p><b>Channel 1 Compression Complete Interrupt Enable Bit.</b> Writing a one to this bit enable the CRC fail interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).</p> <p>User and Privileged mode read</p> <p>0 Compression Complete Interrupt disable</p> <p>1 Compression Complete Interrupt enable</p> <p>User and Privileged mode write</p> <p>0 Has no effect</p> <p>1 Compression Complete Interrupt enable</p>

### 14.4.5 CRC Interrupt Enable Reset Register (CRC\_INTR)

**Figure 14-14. CRC Interrupt Enable Reset Register (CRC\_INTR) [offset = 20h]**

31	29	28	27	26	25	24
Reserved		CH4_TIMEOUT ENR	CH4_UNDERE NR	CH4_OVEREN R	CH4_CRCFAIL ENR	CH4_CCITENR
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	21	20	19	18	17	16
Reserved		CH3_TIMEOUT ENR	CH3_UNDERE NR	CH3_OVEREN R	CH3_CRCFAIL ENR	CH3_CCITENR
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	13	12	11	10	9	8
Reserved		CH2_TIMEOUT ENR	CH2_UNDERE NR	CH2_OVEREN R	CH2_CRCFAIL ENR	CH2_CCITENR
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	5	4	3	2	1	0
Reserved		CH1_TIMEOUT ENR	CH1_UNDERE NR	CH1_OVEREN R	CH1_CRCFAIL ENR	CH1_CCITENR
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

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

**Table 14-9. CRC Interrupt Enable Reset Register (CRC\_INTR) Field Descriptions**

Bit	Field	Value	Description
31-29	Reserved	0	Read returns 0. Writes have no effect.
28	CH4_TIMEOUTENR	0 1 0 1	<p><b>Channel 4 Timeout Interrupt Enable Bit.</b> Writing a one to this bit disable the timeout interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).</p> <p>User and Privileged mode read</p> <p>0 Timeout Interrupt disable</p> <p>1 Timeout Interrupt enable</p> <p>User and Privileged mode write</p> <p>0 Has no effect</p> <p>1 Timeout Interrupt disable</p>
27	CH4_UNDERENR	0 1 0 1	<p><b>Channel 4 Underrun Interrupt Enable Bit.</b> Writing a one to this bit disable the underrun interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).</p> <p>User and Privileged mode read</p> <p>0 Underrun Interrupt disable</p> <p>1 Underrun Interrupt enable</p> <p>User and Privileged mode write</p> <p>0 Has no effect</p> <p>1 Underrun Interrupt disable</p>
26	CH4_OVERENR	0 1 0 1	<p><b>Channel 4 Overrun Interrupt Enable Bit.</b> Writing a one to this bit disable the overrun interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).</p> <p>User and Privileged mode read</p> <p>0 Overrun Interrupt disable</p> <p>1 Overrun Interrupt enable</p> <p>User and Privileged mode write</p> <p>0 Has no effect</p> <p>1 Overrun Interrupt disable</p>

**Table 14-9. CRC Interrupt Enable Reset Register (CRC\_INTR) Field Descriptions (continued)**

Bit	Field	Value	Description
25	CH4_CRCFALENR	  0 1  0 1	<p><b>Channel 4 CRC Fail Interrupt Enable Bit.</b> Writing a one to this bit disable the CRC fail interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).</p> <p>User and Privileged mode read</p> <p>CRC Fail Interrupt disable</p> <p>CRC Fail Interrupt enable</p> <p>User and Privileged mode write</p> <p>Has no effect</p> <p>CRC Fail Interrupt disable</p>
24	CH4_CCITENR	  0 1  0 1	<p><b>Channel 4 Compression Complete Interrupt Enable Bit.</b> Writing a one to this bit disable the CRC fail interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).</p> <p>User and Privileged mode read</p> <p>Compression Complete Interrupt disable</p> <p>Compression Complete Interrupt enable</p> <p>User and Privileged mode write</p> <p>Has no effect</p> <p>Compression Complete Interrupt disable</p>
23-21	Reserved	0	Read returns 0. Writes have no effect.
20	CH3_TIMEOUTENR	  0 1  0 1	<p><b>Channel 3 Timeout Interrupt Enable Bit.</b> Writing a one to this bit disable the timeout interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).</p> <p>User and Privileged mode read</p> <p>Timeout Interrupt disable</p> <p>Timeout Interrupt enable</p> <p>User and Privileged mode write</p> <p>Has no effect</p> <p>Timeout Interrupt disable</p>
19	CH3_UNDERENR	  0 1  0 1	<p><b>Channel 3 Underrun Interrupt Enable Bit.</b> Writing a one to this bit disable the underrun interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).</p> <p>User and Privileged mode read</p> <p>Underrun Interrupt disable</p> <p>Underrun Interrupt enable</p> <p>User and Privileged mode write</p> <p>Has no effect</p> <p>Underrun Interrupt disable</p>
18	CH3_OVERENR	  0 1  0 1	<p><b>Channel 3 Overrun Interrupt Enable Bit.</b> Writing a one to this bit disable the overrun interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).</p> <p>User and Privileged mode read</p> <p>Overrun Interrupt disable</p> <p>Overrun Interrupt enable</p> <p>User and Privileged mode write</p> <p>Has no effect</p> <p>Overrun Interrupt disable</p>



**Table 14-9. CRC Interrupt Enable Reset Register (CRC\_INTR) Field Descriptions (continued)**

Bit	Field	Value	Description
17	CH3_CRCFALENR	<p>User and Privileged mode read</p> <p>0 CRC Fail Interrupt disable</p> <p>1 CRC Fail Interrupt enable</p> <p>User and Privileged mode write</p> <p>0 Has no effect</p> <p>1 CRC Fail Interrupt disable</p>	<p><b>Channel 3 CRC Fail Interrupt Enable Bit.</b> Writing a one to this bit disable the CRC fail interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).</p>
16	CH3_CCITENR	<p>User and Privileged mode read</p> <p>0 Compression Complete Interrupt disable</p> <p>1 Compression Complete Interrupt enable</p> <p>User and Privileged mode write</p> <p>0 Has no effect</p> <p>1 Compression Complete Interrupt disable</p>	<p><b>Channel 3 Compression Complete Interrupt Enable Bit.</b> Writing a one to this bit disable the CRC fail interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).</p>
15-13	Reserved	0	Read returns 0. Writes have no effect.
12	CH2_TIMEOUTENR	<p>User and Privileged mode read</p> <p>0 Timeout Interrupt disable</p> <p>1 Timeout Interrupt enable</p> <p>User and Privileged mode write</p> <p>0 Has no effect</p> <p>1 Timeout Interrupt disable</p>	<p><b>Channel 2 Timeout Interrupt Enable Bit.</b> Writing a one to this bit disable the timeout interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).</p>
11	CH2_UNDERENR	<p>User and Privileged mode read</p> <p>0 Underrun Interrupt disable</p> <p>1 Underrun Interrupt enable</p> <p>User and Privileged mode write</p> <p>0 Has no effect</p> <p>1 Underrun Interrupt disable</p>	<p><b>Channel 2 Underrun Interrupt Enable Bit.</b> Writing a one to this bit disable the underrun interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).</p>
10	CH2_OVERENR	<p>User and Privileged mode read</p> <p>0 Overrun Interrupt disable</p> <p>1 Overrun Interrupt enable</p> <p>User and Privileged mode write</p> <p>0 Has no effect</p> <p>1 Overrun Interrupt disable</p>	<p><b>Channel 2 Overrun Interrupt Enable Bit.</b> Writing a one to this bit disable the overrun interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).</p>

**Table 14-9. CRC Interrupt Enable Reset Register (CRC\_INTR) Field Descriptions (continued)**

Bit	Field	Value	Description
9	CH2_CRCFALENR	  0 1  0 1	<p><b>Channel 2 CRC Fail Interrupt Enable Bit.</b> Writing a one to this bit disable the CRC fail interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).</p> <p>User and Privileged mode read</p> <p>CRC Fail Interrupt disable</p> <p>CRC Fail Interrupt enable</p> <p>User and Privileged mode write</p> <p>Has no effect</p> <p>CRC Fail Interrupt disable</p>
8	CH2_CCITENR	  0 1  0 1	<p><b>Channel 2 Compression Complete Interrupt Enable Bit.</b> Writing a one to this bit disable the CRC fail interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).</p> <p>User and Privileged mode read</p> <p>Compression Complete Interrupt disable</p> <p>Compression Complete Interrupt enable</p> <p>User and Privileged mode write</p> <p>Has no effect</p> <p>Compression Complete Interrupt disable</p>
7-5	Reserved	0	Read returns 0. Writes have no effect.
4	CH1_TIMEOUTENR	  0 1  0 1	<p><b>Channel 1 Timeout Interrupt Enable Bit.</b> Writing a one to this bit disable the timeout interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).</p> <p>User and Privileged mode read</p> <p>Timeout Interrupt disable</p> <p>Timeout Interrupt enable</p> <p>User and Privileged mode write</p> <p>Has no effect</p> <p>Timeout Interrupt disable</p>
3	CH1_UNDERENR	  0 1  0 1	<p><b>Channel 1 Underrun Interrupt Enable Bit.</b> Writing a one to this bit disable the underrun interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).</p> <p>User and Privileged mode read</p> <p>Underrun Interrupt disable</p> <p>Underrun Interrupt enable</p> <p>User and Privileged mode write</p> <p>Has no effect</p> <p>Underrun Interrupt disable</p>
2	CH1_OVERENR	  0 1  0 1	<p><b>Channel 1 Overrun Interrupt Enable Bit.</b> Writing a one to this bit disable the overrun interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).</p> <p>User and Privileged mode read</p> <p>Overrun Interrupt disable</p> <p>Overrun Interrupt enable</p> <p>User and Privileged mode write</p> <p>Has no effect</p> <p>Overrun Interrupt disable</p>

**Table 14-9. CRC Interrupt Enable Reset Register (CRC\_INTR) Field Descriptions (continued)**

Bit	Field	Value	Description
1	CH1_CRCFAILENR		<p><b>Channel 1 CRC Fail Interrupt Enable Bit.</b> Writing a one to this bit disable the CRC fail interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).</p> <p>User and Privileged mode read</p> <p>0 CRC Fail Interrupt disable</p> <p>1 CRC Fail Interrupt enable</p> <p>User and Privileged mode write</p> <p>0 Has no effect</p> <p>1 CRC Fail Interrupt disable</p>
0	CH1_CCITENR		<p><b>Channel 1 Compression Complete Interrupt Enable Bit.</b> Writing a one to this bit disable the CRC fail interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).</p> <p>User and Privileged mode read</p> <p>0 Compression Complete Interrupt disable</p> <p>1 Compression Complete Interrupt enable</p> <p>User and Privileged mode write</p> <p>0 Has no effect</p> <p>1 Compression Complete Interrupt disable</p>

### 14.4.6 CRC Interrupt Status Register (CRC\_STATUS)

**Figure 14-15. CRC Interrupt Status Register (CRC\_STATUS) [offset = 28h]**

31	30	29	28	27	26	25	24
Reserved			CH4_TIMEOUT	CH4_UNDER	CH4_OVER	CH4_CRCFAIL	CH4_CCIT
R-0			R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	21	20	19	18	17	16	
Reserved			CH3_TIMEOUT	CH3_UNDER	CH3_OVER	CH3_CRCFAIL	CH3_CCIT
R-0			R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	13	12	11	10	9	8	
Reserved			CH2_TIMEOUT	CH2_UNDER	CH2_OVER	CH2_CRCFAIL	CH2_CCIT
R-0			R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	5	4	3	2	1	0	
Reserved			CH1_TIMEOUT	CH1_UNDER	CH1_OVER	CH1_CRCFAIL	CH1_CCIT
R-0			R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

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

**Table 14-10. CRC Interrupt Status Register (CRC\_STATUS) Field Descriptions**

Bit	Field	Value	Description
31-29	Reserved	0	Read returns 0. Writes have no effect.
28	CH4_TIMEOUT	0 1	<b>Channel 4 CRC Timeout Status Flag.</b> This bit is cleared by writing a '1' to it only. Writing '0' has no effect. This bit is set in both AUTO and Semi-CPU mode 0 No timeout interrupt is active 1 Timeout interrupt is active
27	CH4_UNDER	0 1	<b>Channel 4 CRC Underrun Status Flag.</b> This bit is cleared by writing a '1' to it only. Writing '0' has no effect. This bit is set in AUTO mode only. 0 No Underrun Interrupt is active 1 Underrun Interrupt is active
26	CH4_OVER	0 1	<b>Channel 4 CRC Overrun Status Flag.</b> This bit is cleared by writing a '1' to it only. Writing '0' has no effect. This bit is set in either AUTO or Semi-CPU mode. 0 No Overrun Interrupt is active 1 Overrun Interrupt is active
25	CH4_CRCFAIL	0 1	<b>Channel 4 CRC Compare Fail Status Flag.</b> This bit is cleared by writing a '1' to it only. Writing '0' has no effect. This bit is set in AUTO mode only. 0 No CRC Fail Interrupt is active 1 CRC Fail Interrupt is active
24	CH4_CCIT	0 1	<b>Channel 4 CRC Pattern Compression Complete Status Flag.</b> This bit is cleared by writing a '1' to it only. Writing '0' has no effect. This bit is only set in Semi-CPU mode. 0 Has no effect 1 Compression Complete Interrupt disable
23-21	Reserved	0	Read returns 0. Writes have no effect.
20	CH3_TIMEOUT	0 1	<b>Channel 3 CRC Timeout Status Flag.</b> This bit is cleared by writing a '1' to it only. Writing '0' has no effect. This bit is set in both AUTO and Semi-CPU mode 0 No timeout interrupt is active 1 Timeout interrupt is active
19	CH3_UNDER	0 1	<b>Channel 3 CRC Underrun Status Flag.</b> This bit is cleared by writing a '1' to it only. Writing '0' has no effect. This bit is set in AUTO mode only. 0 No Underrun Interrupt is active 1 Underrun Interrupt is active

**Table 14-10. CRC Interrupt Status Register (CRC\_STATUS) Field Descriptions (continued)**

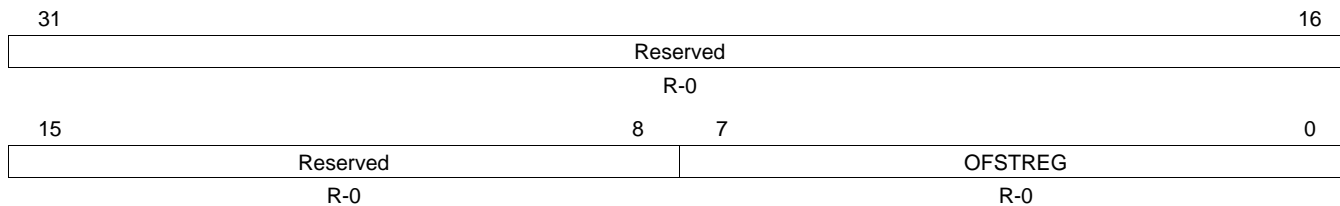
Bit	Field	Value	Description
18	CH3_OVER		<b>Channel 3 CRC Overrun Status Flag.</b> This bit is cleared by writing a '1' to it only. Writing '0' has no effect. This bit is set in either AUTO or Semi-CPU mode.
		0	No Overrun Interrupt is active
		1	Overrun Interrupt is active
		17	CH3_CRCFAIL
0	No CRC Fail Interrupt is active		
		1	CRC Fail Interrupt is active
		16	CH3_CCIT
0	Has no effect		
		1	Compression Complete Interrupt disable
		15-13	Reserved
12	CH2_TIMEOUT		<b>Channel 2 CRC Timeout Status Flag.</b> This bit is cleared by writing a '1' to it only. Writing '0' has no effect. This bit is set in both AUTO and Semi-CPU mode
		0	No timeout interrupt is active
		1	Timeout interrupt is active
		11	CH2_UNDER
0	No Underrun Interrupt is active		
		1	Underrun Interrupt is active
		10	CH2_OVER
0	No Overrun Interrupt is active		
		1	Overrun Interrupt is active
		9	CH2_CRCFAIL
0	No CRC Fail Interrupt is active		
		1	CRC Fail Interrupt is active
		8	CH2_CCIT
0	Has no effect		
		1	Compression Complete Interrupt disable
		7-5	Reserved
4	CH1_TIMEOUT		<b>Channel 1 CRC Timeout Status Flag.</b> Writing a one to this bit enable the timeout interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
		0	Has no effect
		1	Timeout Interrupt disable
		3	CH1_UNDER
0	Has no effect		
		1	Underrun Interrupt disable
		2	CH1_OVER
0	Has no effect		
		1	Overrun Interrupt disable
		1	CH1_CRCFAIL
0	Has no effect		
		1	CRC Fail Interrupt disable

**Table 14-10. CRC Interrupt Status Register (CRC\_STATUS) Field Descriptions (continued)**

Bit	Field	Value	Description
0	CH1_CCIT		<b>Channel 4 CRC Pattern Compression Complete Status Flag.</b> Writing a one to this bit disable the Compression Complete interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
		0	Has no effect
		1	Compression Complete Interrupt disable

### 14.4.7 CRC Interrupt Offset (CRC\_INT\_OFFSET\_REG)

Figure 14-16. CRC Interrupt Offset (CRC\_INT\_OFFSET\_REG) [offset = 30h]



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

Table 14-11. CRC Interrupt Offset (CRC\_INT\_OFFSET\_REG) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Read returns 0. Writes have no effect.
7-0	OFSTREG		<b>CRC Interrupt Offset.</b> This register indicates the highest priority pending interrupt vector address. Reading the offset register automatically clear the respective interrupt flag. Please reference <a href="#">Table 14-3</a> for details.

### 14.4.8 CRC Busy Register (CRC\_BUSY)

**Figure 14-17. CRC Busy Register (CRC\_BUSY) [offset = 38h]**

31	25	24	23	17	16
Reserved	CH4_BUSY		Reserved	CH3_BUSY	
R-0	R-0		R-0	R-0	
15	9	8	7	1	0
Reserved	CH2_BUSY		Reserved	CH1_BUSY	
R-0	R-0		R-0	R-0	

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

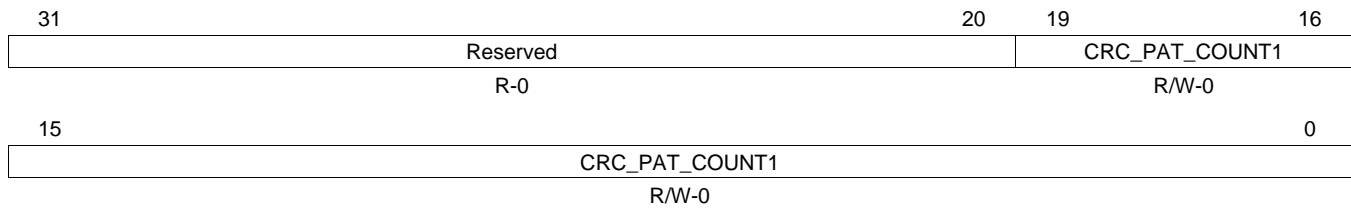
**Table 14-12. CRC Busy Register (CRC\_BUSY) Field Descriptions**

Bit	Field	Value	Description
31-25	Reserved	0	Read returns 0. Writes have no effect.
24	CH4_BUSY		<b>CH4_BUSY.</b> During AUTO or Semi-CPU mode, the busy flag is set when the first data pattern of the block is compressed and remains set until the the last data pattern of the block is compressed. The flag is cleared when the last data pattern of the block is compressed.
23-17	Reserved		Read returns 0. Writes have no effect.
16	CH3_BUSY		<b>CH3_BUSY.</b> During AUTO or Semi-CPU mode, the busy flag is set when the first data pattern of the block is compressed and remains set until the the last data pattern of the block is compressed. The flag is cleared when the last data pattern of the block is compressed.
15-9	Reserved		Read returns 0. Writes have no effect.
8	CH2_BUSY		<b>CH2_BUSY.</b> During AUTO or Semi-CPU mode, the busy flag is set when the first data pattern of the block is compressed and remains set until the the last data pattern of the block is compressed. The flag is cleared when the last data pattern of the block is compressed.
7-1	Reserved		Read returns 0. Writes have no effect.
0	CH1_BUSY		<b>CH1_BUSY.</b> During AUTO or Semi-CPU mode, the busy flag is set when the first data pattern of the block is compressed and remains set until the the last data pattern of the block is compressed. The flag is cleared when the last data pattern of the block is compressed.



### 14.4.9 CRC Pattern Counter Preload Register 1 (CRC\_PCOUNT\_REG1)

Figure 14-18. CRC Pattern Counter Preload Register 1 (CRC\_PCOUNT\_REG1) [offset = 40h]



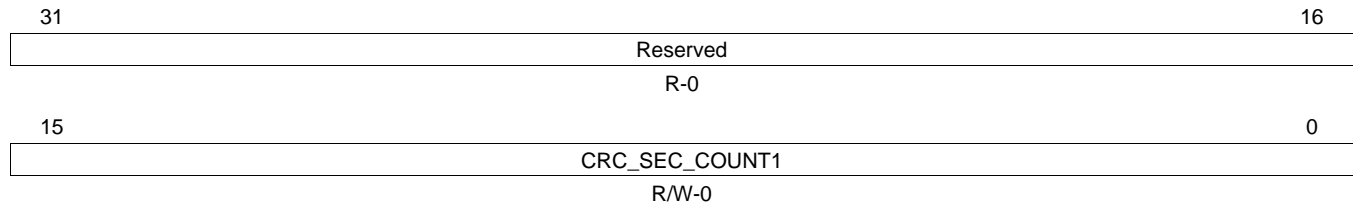
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 14-13. CRC Pattern Counter Preload Register 1 (CRC\_PCOUNT\_REG1) Field Descriptions

Bit	Field	Value	Description
31-20	Reserved	0	Read returns 0. Writes have no effect.
19-0	CRC_PAT_COUNT 1		<b>Channel 1 Pattern Counter Preload Register.</b> This register contains the number of data patterns in one sector to be compressed before a CRC is performed.

### 14.4.10 CRC Sector Counter Preload Register 1 (CRC\_SCOUNT\_REG1)

**Figure 14-19. CRC Sector Counter Preload Register 1 (CRC\_SCOUNT\_REG1) [offset = 44h]**



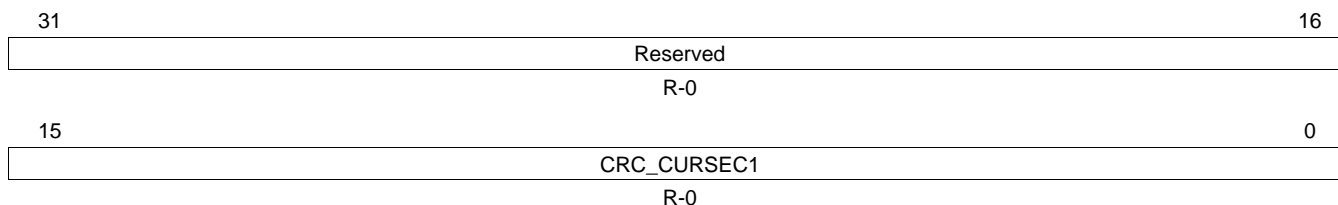
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-14. CRC Sector Counter Preload Register 1 (CRC\_SCOUNT\_REG1) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	CRC_SEC_COUNT 1		<b>Channel 1 Sector Counter Preload Register.</b> This register contains the number of sectors in one block of memory.

### 14.4.11 CRC Current Sector Register 1 (CRC\_CURSEC\_REG1)

Figure 14-20. CRC Current Sector Preload Register 1 (CRC\_CURSEC\_REG1) [offset = 48h]



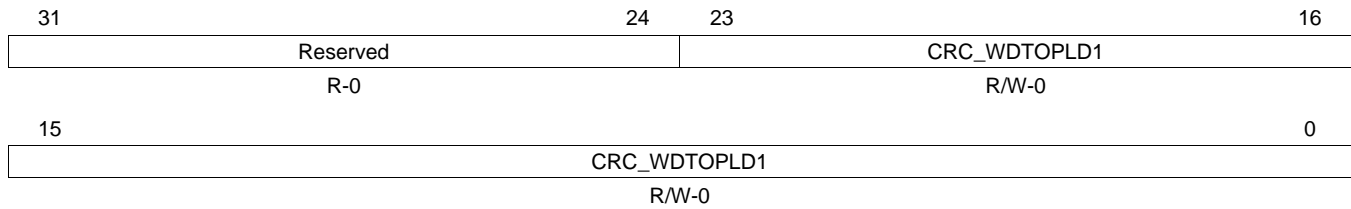
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 14-15. CRC Current Sector Register 1 (CRC\_CURSEC\_REG1) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	CRC_CURSEC1		<b>Channel 1 Current Sector ID Register.</b> In AUTO mode, this register contains the current sector number of which the signature verification fails. The sector counter is a free running up counter. When a sector fails, the erroneous sector number is logged into current sector ID register and the CRC fail interrupt is generated. The sector ID register is frozen until it is read and the CRC fail status bit is cleared by CPU. While it is frozen, it does not capture another erroneous sector number. When this condition happens, an overrun interrupt is generated instead. Once the register is read and the CRC fail interrupt flag is cleared it can capture new erroneous sector number. In Semi-CPU mode, this register is used to indicate the sector number for which the compression complete has last happened.

### 14.4.12 CRC Channel 1 Watchdog Timeout Preload Register A (CRC\_WDTPLD1)

**Figure 14-21. CRC Channel 1 Watchdog Timeout Preload Register A (CRC\_WDTPLD1) [offset = 4Ch]**



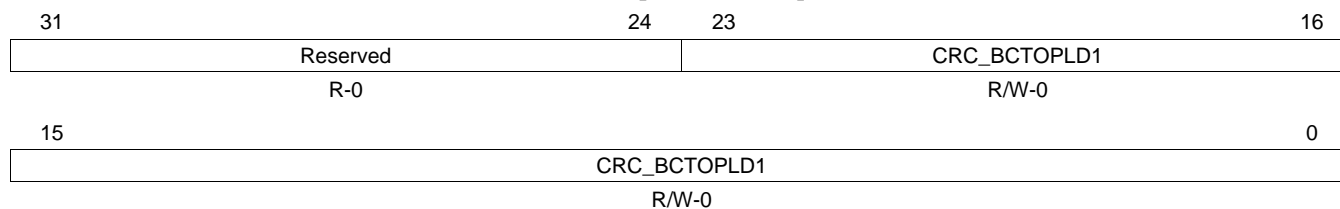
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-16. CRC Channel 1 Watchdog Timeout Preload Register A (CRC\_WDTPLD1) Field Descriptions**

Bit	Field	Value	Description
31-24	Reserved	0	Read returns 0. Writes have no effect.
23-0	CRC_WDTPLD1		<b>Channel 1 Watchdog Timeout Counter Preload Register.</b> This register contains the number of clock cycles within which the DMA must transfer the next block of data patterns. In Semi-CPU mode, this register is used to indicate the sector number for which the compression complete has last happened.

### 14.4.13 CRC Channel 1 Block Complete Timeout Preload Register B (CRC\_BCTOPLD1)

**Figure 14-22. CRC Channel 1 Block Complete Timeout Preload Register B (CRC\_BCTOPLD1)  
[offset = 50h]**



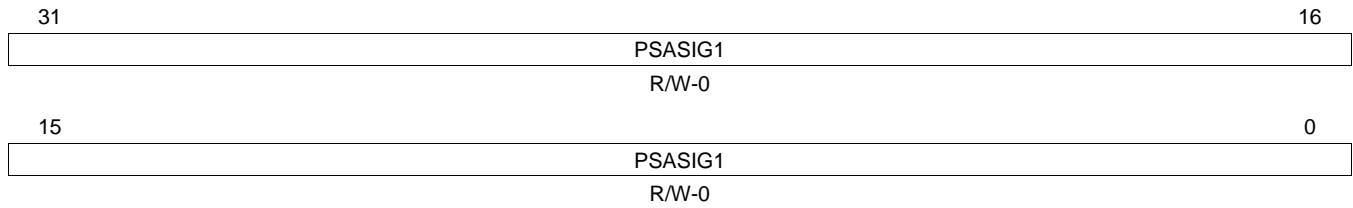
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-17. CRC Channel 1 Block Complete Timeout Preload Register B (CRC\_BCTOPLD1) Field Descriptions**

Bit	Field	Value	Description
31-24	Reserved	0	Read returns 0. Writes have no effect.
23-0	CRC_BCTOPLD1		<b>Channel 1 Block Complete Timeout Counter Preload Register.</b> This register contains the number of clock cycles within which the CRC for an entire block needs to complete before a timeout interrupt is generated.

**14.4.14 Channel 1 PSA Signature Low Register (PSA\_SIGREGL1)**

**Figure 14-23. Channel 1 PSA Signature Low Register (PSA\_SIGREGL1) [offset = 60h]**



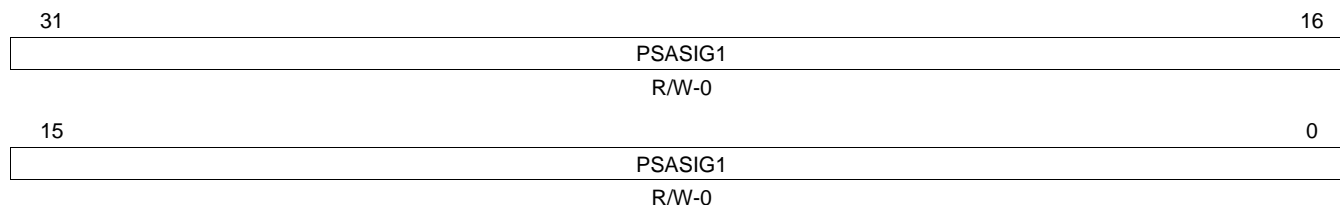
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-18. Channel 1 PSA Signature Low Register (PSA\_SIGREGL1) Field Descriptions**

Bit	Field	Value	Description
31-0	PSASIG1		<b>Channel 1 PSA Signature Low Register.</b> This register contains the value stored at PSASIG1[31:0] register.

### 14.4.15 Channel 1 PSA Signature High Register (PSA\_SIGREGH1)

Figure 14-24. Channel 1 PSA Signature High Register (PSA\_SIGREGH1) [offset = 64h]



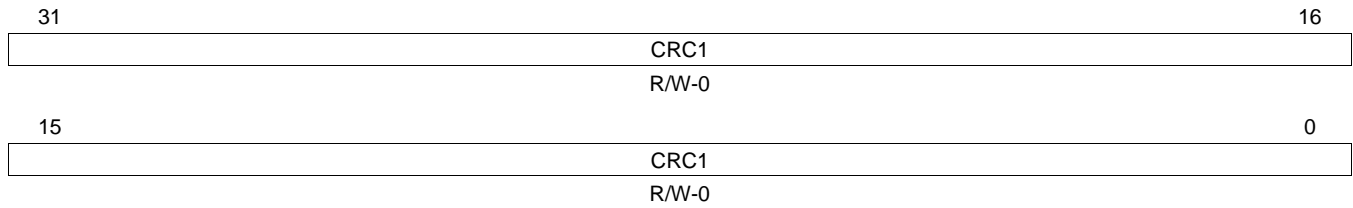
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 14-19. Channel 1 PSA Signature High Register (PSA\_SIGREGH1) Field Descriptions

Bit	Field	Value	Description
31-0	PSASIG1		<b>Channel 1 PSA Signature Low Register.</b> This register contains the value stored at PSASIG1[63:32] register.

**14.4.16 Channel 1 CRC Value Low Register (CRC\_REGL1)**

**Figure 14-25. Channel 1 CRC Value Low Register (CRC\_REGL1) [offset = 68h]**



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

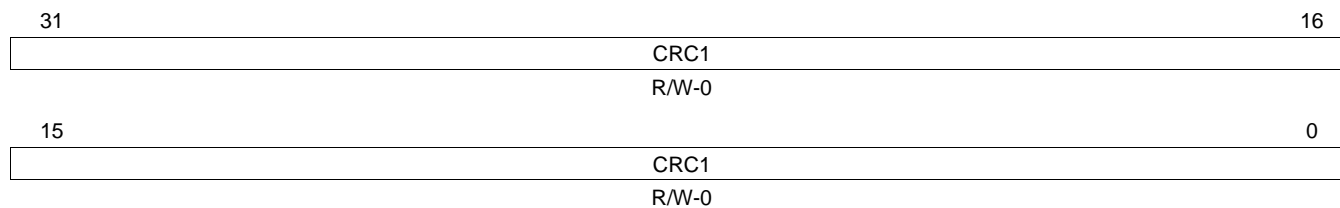
**Table 14-20. Channel 1 CRC Value Low Register (CRC\_REGL1) Field Descriptions**

Bit	Field	Value	Description
31-0	CRC1		<b>Channel 1 CRC Value Low Register.</b> This register contains the current known good signature value stored at CRC1[31:0] register.



### 14.4.17 Channel 1 CRC Value High Register (CRC\_REGH1)

**Figure 14-26. Channel 1 CRC Value High Register (CRC\_REGH1) [offset = 6Ch]**



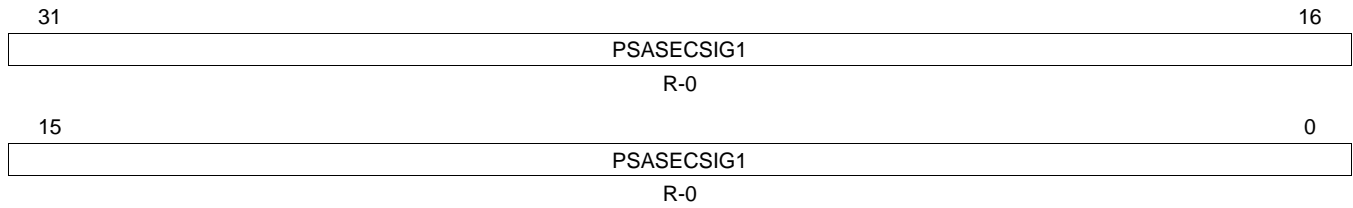
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-21. Channel 1 CRC Value High Register (CRC\_REGH1) Field Descriptions**

Bit	Field	Value	Description
31-0	CRC1		<b>Channel 1 CRC Value Low Register.</b> This register contains the current known good signature value stored at CRC1[63:32] register.

### 14.4.18 Channel 1 PSA Sector Signature Low Register (PSA\_SECSIGREGL1)

**Figure 14-27. Channel 1 PSA Sector Signature Low Register (PSA\_SECSIGREGL1) [offset = 70h]**



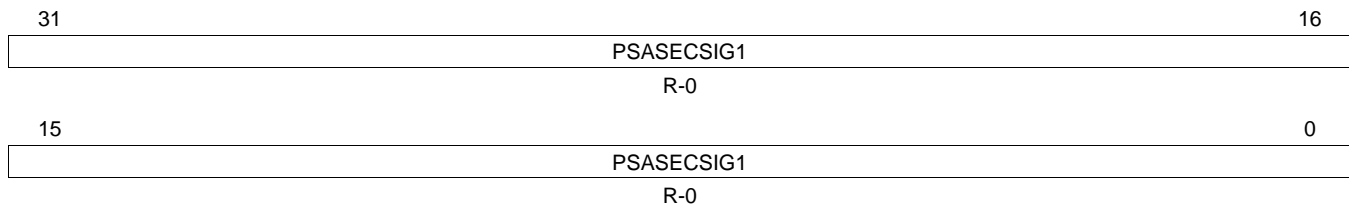
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-22. Channel 1 PSA Sector Signature Low Register (PSA\_SECSIGREGL1) Field Descriptions**

Bit	Field	Value	Description
31-0	PSASECSIG1		<b>Channel 1 PSA Sector Signature Low Register.</b> This register contains the value stored at PSASECSIG1[31:0] register.

### 14.4.19 Channel 1 PSA Sector Signature High Register (PSA\_SECSIGREGH1)

Figure 14-28. Channel 1 PSA Sector Signature High Register (PSA\_SECSIGREGH1) [offset = 74h]



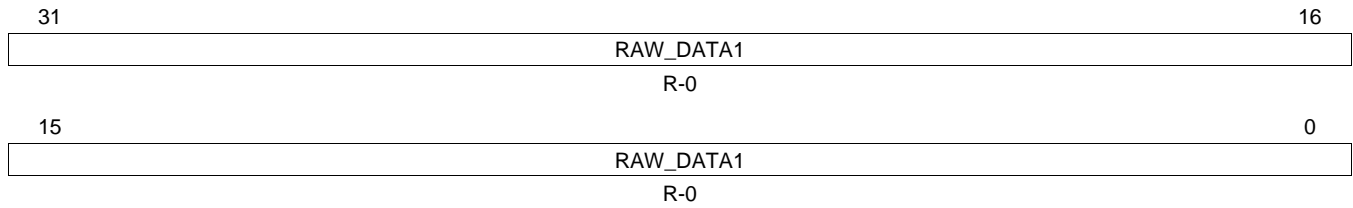
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 14-23. Channel 1 PSA Sector Signature High Register (PSA\_SECSIGREGH1) Field Descriptions

Bit	Field	Value	Description
31-0	PSASECSIG1		<b>Channel 1 PSA Sector Signature Low Register.</b> This register contains the value stored at PSASECSIG1[63:32] register.

**14.4.20 Channel 1 Raw Data Low Register (RAW\_DATAREGL1)**

**Figure 14-29. Channel 1 Raw Data Low Register (RAW\_DATAREGL1) [offset = 78h]**



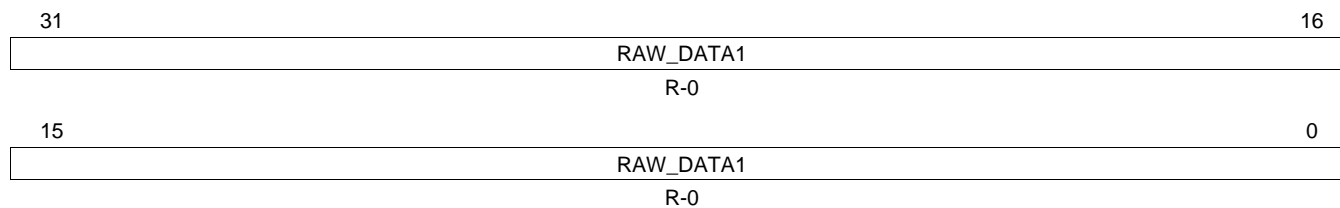
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-24. Channel 1 Raw Data Low Register (RAW\_DATAREGL1) Field Descriptions**

Bit	Field	Value	Description
31-0	RAW_DATA1		<b>Channel 1 Raw Data Low Register.</b> This register contains bit 31:0 of the un-compressed raw data.

### 14.4.21 Channel 1 Raw Data High Register (RAW\_DATAREGH1)

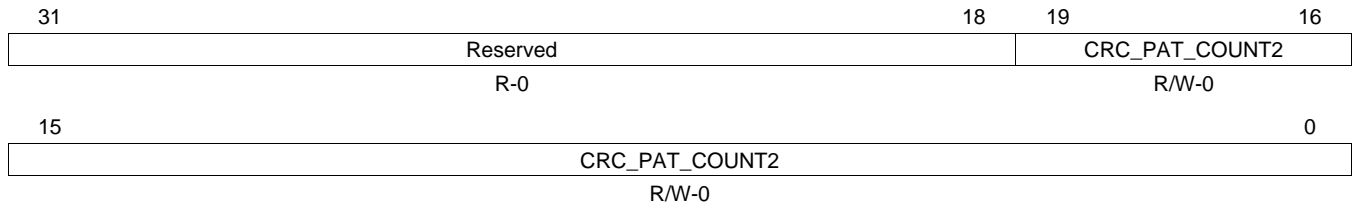
Figure 14-30. Channel 1 Raw Data High Register (RAW\_DATAREGH1) [offset = 7Ch]



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

Table 14-25. Channel 1 Raw Data High Register (RAW\_DATAREGH1) Field Descriptions

Bit	Field	Value	Description
31-0	RAW_DATA1		<b>Channel 1 Raw Data Low Register.</b> This register contains bit 63:32 of the un-compressed raw data.

**14.4.22 CRC Pattern Counter Preload Register 2 (CRC\_PCOUNT\_REG2)**
**Figure 14-31. CRC Pattern Counter Preload Register 2 (CRC\_PCOUNT\_REG2) [offset = 80h]**


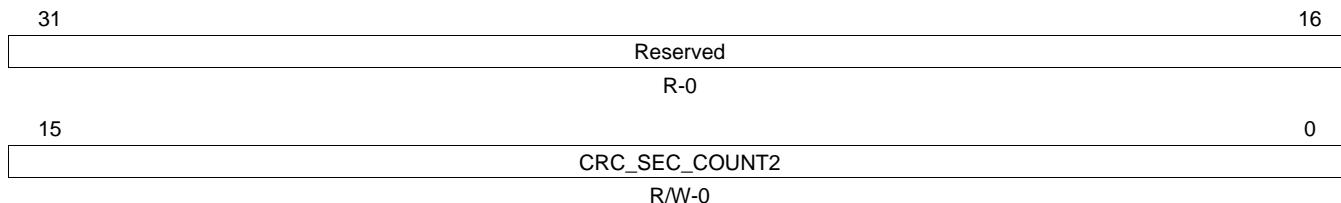
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-26. CRC Pattern Counter Preload Register 2 (CRC\_PCOUNT\_REG2) Field Descriptions**

Bit	Field	Value	Description
31-20	Reserved	0	Read returns 0. Writes have no effect.
19-0	CRC_PAT_COUNT 2		<b>Channel 2 Pattern Counter Preload Register.</b> This register contains the number of data patterns in one sector to be compressed before a CRC is performed.

### 14.4.23 CRC Sector Counter Preload Register 2 (CRC\_SCOUNT\_REG2)

Figure 14-32. CRC Sector Counter Preload Register 2 (CRC\_SCOUNT\_REG2) [offset = 84h]



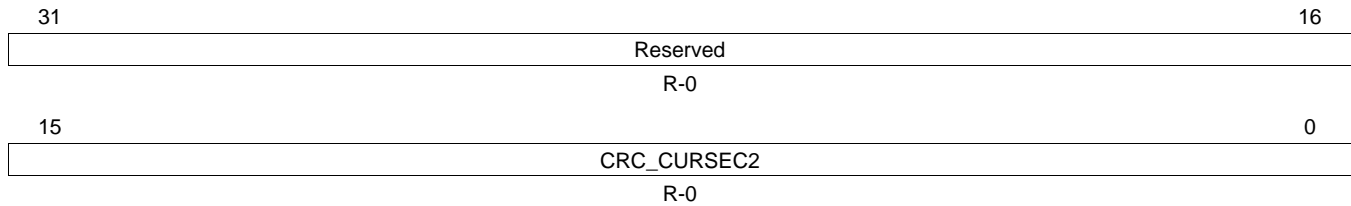
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 14-27. CRC Sector Counter Preload Register 2 (CRC\_SCOUNT\_REG2) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	CRC_SEC_COUNT2		<b>Channel 2 Sector Counter Preload Register.</b> This register contains the number of sectors in one block of memory.

**14.4.24 CRC Current Sector Register 2 (CRC\_CURSEC\_REG2)**

**Figure 14-33. CRC Current Sector Register 2 (CRC\_CURSEC\_REG2) [offset = 88h]**



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

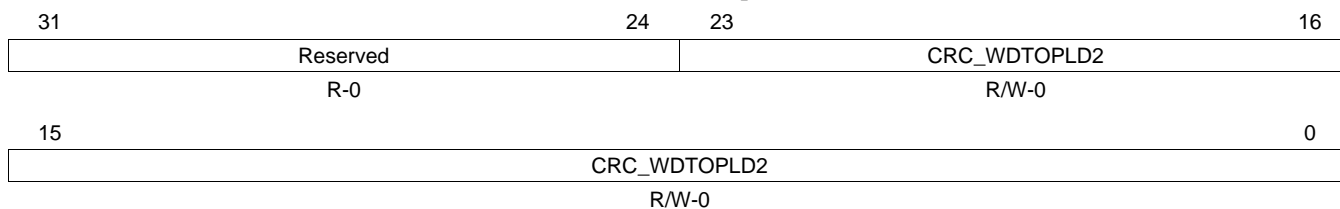
**Table 14-28. CRC Current Sector Register 2 (CRC\_CURSEC\_REG2) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	CRC_CURSEC2		<b>Channel 2 Current Sector ID Register.</b> In AUTO mode, this register contains the current sector number of which the signature verification fails. The sector counter is a free running up counter. When a sector fails, the erroneous sector number is logged into current sector ID register and the CRC fail interrupt is generated. The sector ID register is frozen until it is read and the CRC fail status bit is cleared by CPU. While it is frozen, it does not capture another erroneous sector number. When this condition happens, an overrun interrupt is generated instead. Once the register is read and the CRC fail interrupt flag is cleared it can capture new erroneous sector number. In Semi-CPU mode, this register is used to indicate the sector number for which the compression complete has last happened.



### 14.4.25 CRC Channel 2 Watchdog Timeout Preload Register A (CRC\_WDTPLD2)

**Figure 14-34. CRC Channel 2 Watchdog Timeout Preload Register A (CRC\_WDTPLD2) [offset = 8Ch]**



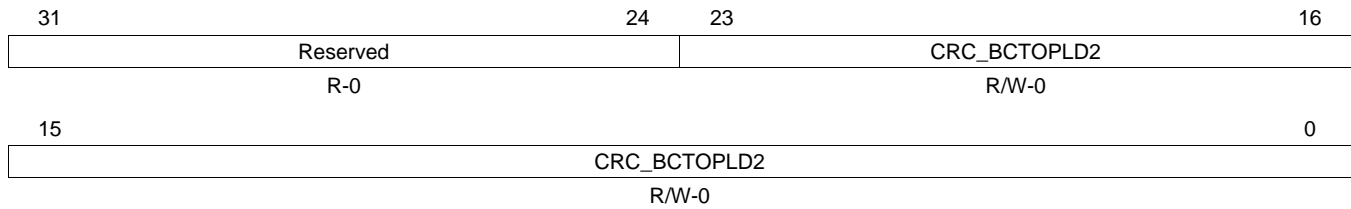
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-29. CRC Channel 2 Watchdog Timeout Preload Register A (CRC\_WDTPLD2) Field Descriptions**

Bit	Field	Value	Description
31-24	Reserved	0	Read returns 0. Writes have no effect.
23-0	CRC_WDTPLD2		<b>Channel 2 Watchdog Timeout Counter Preload Register.</b> This register contains the number of clock cycles within which the DMA must transfer the next block of data patterns. In Semi-CPU mode, this register is used to indicate the sector number for which the compression complete has last happened.

### 14.4.26 CRC Channel 2 Block Complete Timeout Preload Register B (CRC\_BCTOPLD2)

**Figure 14-35. CRC Channel 2 Block Complete Timeout Preload Register B (CRC\_BCTOPLD2)  
[offset = 90h]**



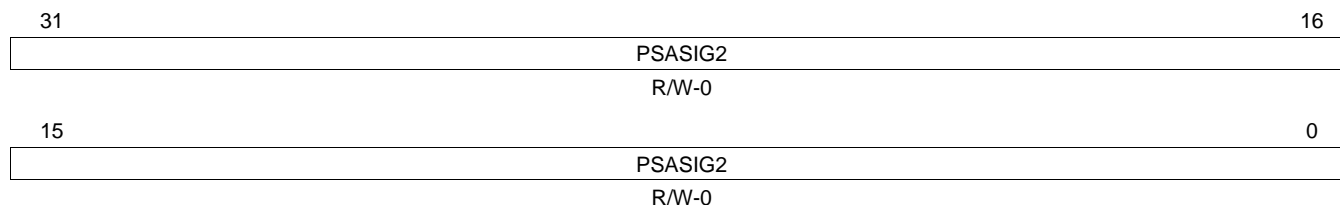
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-30. CRC Channel 2 Block Complete Timeout Preload Register B (CRC\_BCTOPLD2) Field Descriptions**

Bit	Field	Value	Description
31-24	Reserved	0	Read returns 0. Writes have no effect.
23-0	CRC_BCTOPLD2		<b>Channel 2 Block Complete Timeout Counter Preload Register.</b> This register contains the number of clock cycles within which the CRC for an entire block needs to complete before a timeout interrupt is generated.

### 14.4.27 Channel 2 PSA Signature Low Register (PSA\_SIGREGL2)

Figure 14-36. Channel 2 PSA Signature Low Register (PSA\_SIGREGL2) [offset = A0h]



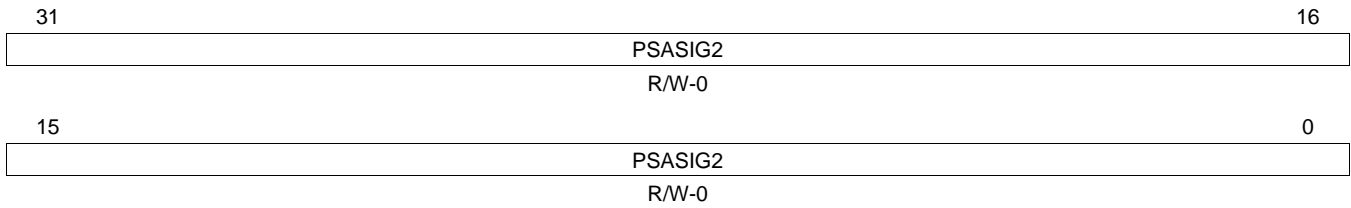
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 14-31. Channel 2 PSA Signature Low Register (PSA\_SIGREGL2) Field Descriptions

Bit	Field	Value	Description
31-0	PSASIG2		<b>Channel 2 PSA Signature Low Register.</b> This register contains the value stored at PSASIG2[31:0] register.

**14.4.28 Channel 2 PSA Signature High Register (PSA\_SIGREGH2)**

**Figure 14-37. Channel 2 PSA Signature High Register (PSA\_SIGREGH2) [offset = A4h]**



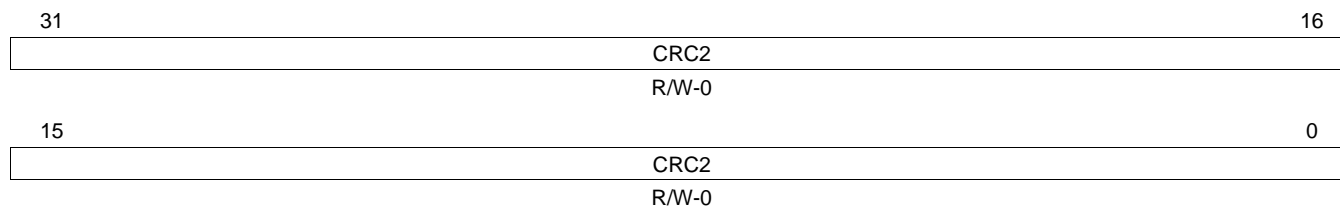
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-32. Channel 2 PSA Signature High Register (PSA\_SIGREGH2) Field Descriptions**

Bit	Field	Value	Description
31-0	PSASIG2		<b>Channel 2 PSA Signature High Register.</b> This register contains the value stored at PSASIG2[63:32] register.

### 14.4.29 Channel 2 CRC Value Low Register (CRC\_REGL2)

**Figure 14-38. Channel 2 CRC Value Low Register (CRC\_REGL2) [offset = A8h]**



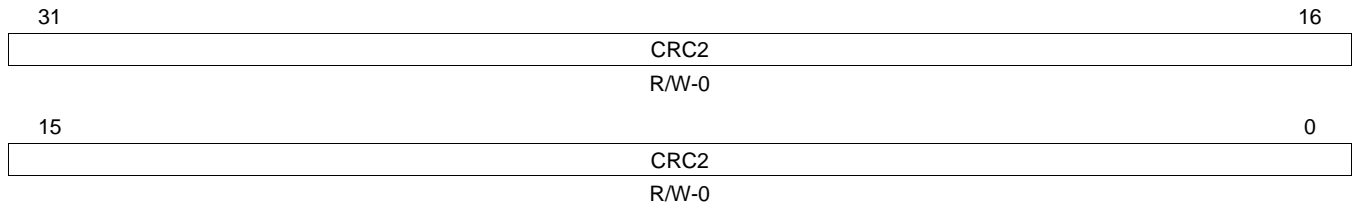
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-33. Channel 2 CRC Value Low Register (CRC\_REGL2) Field Descriptions**

Bit	Field	Value	Description
31-0	CRC2		<b>Channel 2 CRC Value Low Register.</b> This register contains the current known good signature value stored at CRC2[31:0] register.

**14.4.30 Channel 2 CRC Value High Register (CRC\_REGH2)**

**Figure 14-39. Channel 2 CRC Value High Register (CRC\_REGH2) [offset = ACh]**



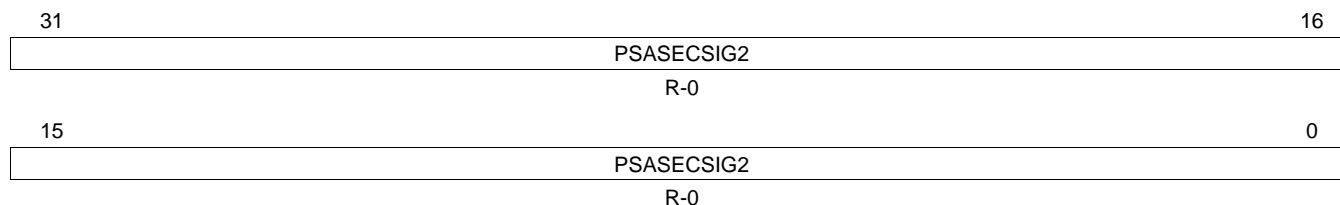
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-34. Channel 2 CRC Value High Register (CRC\_REGH2) Field Descriptions**

Bit	Field	Value	Description
31-0	CRC2		<b>Channel 2 CRC Value High Register.</b> This register contains the current known good signature value stored at CRC2[63:32] register.

### 14.4.31 Channel 2 PSA Sector Signature Low Register (PSA\_SECSIGREGL2)

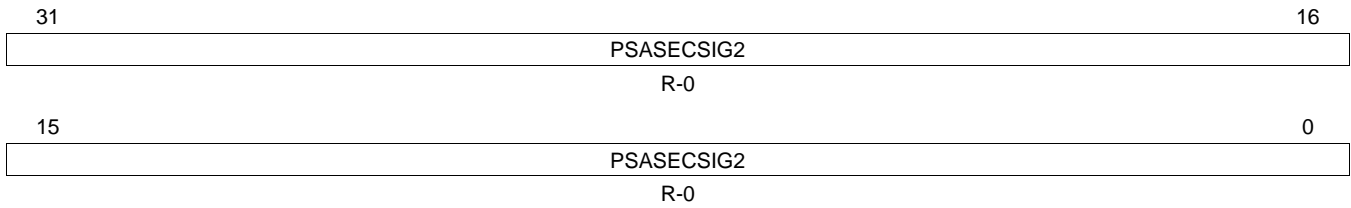
Figure 14-40. Channel 2 PSA Sector Signature Low Register (PSA\_SECSIGREGL2) [offset = B0h]



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

Table 14-35. Channel 2 PSA Sector Signature Low Register (PSA\_SECSIGREGL2) Field Descriptions

Bit	Field	Value	Description
31-0	PSASECSIG2		<b>Channel 2 PSA Sector Signature Low Register.</b> This register contains the value stored at PSASECSIG2[31:0] register.

**14.4.32 Channel 2 PSA Sector Signature High Register (PSA\_SECSIGREGH2)**
**Figure 14-41. Channel 2 PSA Sector Signature High Register (PSA\_SECSIGREGH2) [offset = B4h]**


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

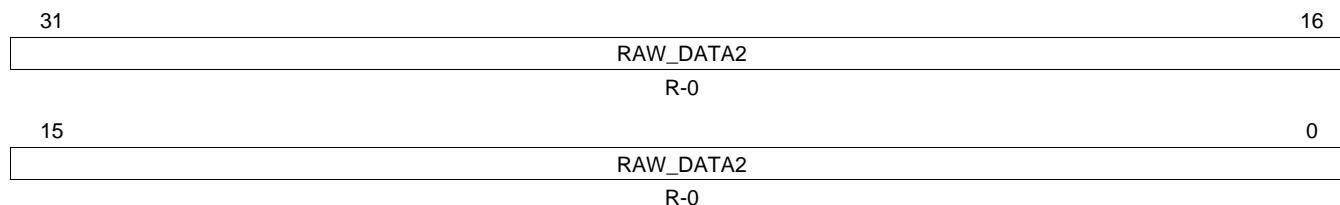
**Table 14-36. Channel 2 PSA Sector Signature High Register (PSA\_SECSIGREGH2) Field Descriptions**

Bit	Field	Value	Description
31-0	PSASECSIG2		<b>Channel 2 PSA Sector Signature High Register.</b> This register contains the value stored at PSASECSIG2[63:32] register.



### 14.4.33 Channel 2 Raw Data Low Register (RAW\_DATAREGL2)

Figure 14-42. Channel 2 Raw Data Low Register (RAW\_DATAREGL2) [offset = B8h]



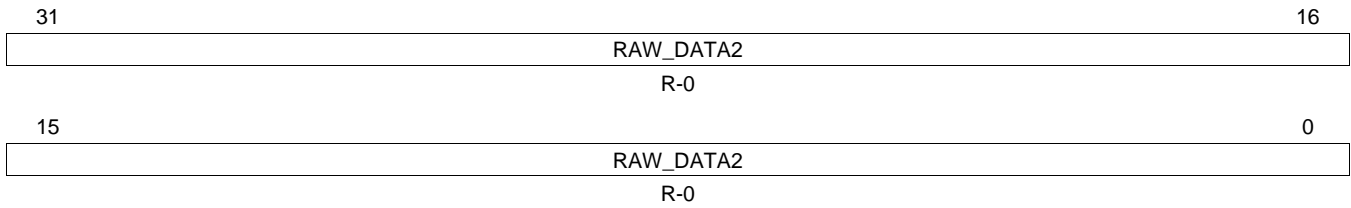
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 14-37. Channel 2 Raw Data Low Register (RAW\_DATAREGL2) Field Descriptions

Bit	Field	Value	Description
31-0	RAW_DATA2		<b>Channel 2 Raw Data Low Register.</b> This register contains bit 31:0 of the un-compressed raw data..

**14.4.34 Channel 2 Raw Data High Register (RAW\_DATAREGH2)**

**Figure 14-43. Channel 2 Raw Data High Register (RAW\_DATAREGH2) [offset = BCh]**



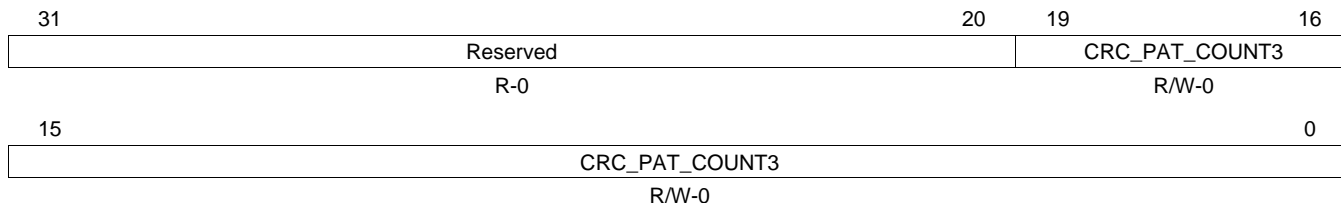
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-38. Channel 2 Raw Data High Register (RAW\_DATAREGH2) Field Descriptions**

Bit	Field	Value	Description
31-0	RAW_DATA2		<b>Channel 2 Raw Data High Register.</b> This register contains bit 63:32 of the un-compressed raw data..

### 14.4.35 CRC Pattern Counter Preload Register 3 (CRC\_PCOUNT\_REG3)

Figure 14-44. CRC Pattern Counter Preload Register 3 (CRC\_PCOUNT\_REG3) [offset = C0h]



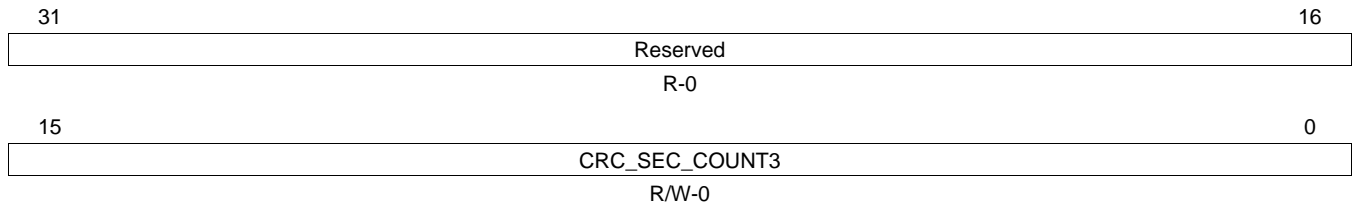
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 14-39. CRC Pattern Counter Preload Register 3 (CRC\_PCOUNT\_REG3) Field Descriptions

Bit	Field	Value	Description
31-20	Reserved	0	Read returns 0. Writes have no effect.
19-0	CRC_PAT_COUNT 3		<b>Channel 3 Pattern Counter Preload Register.</b> This register contains the number of data patterns in one sector to be compressed before a CRC is performed.

### 14.4.36 CRC Sector Counter Preload Register 3 (CRC\_SCOUNT\_REG3)

**Figure 14-45. CRC Sector Counter Preload Register 3 (CRC\_SCOUNT\_REG3) [offset = C4h]**



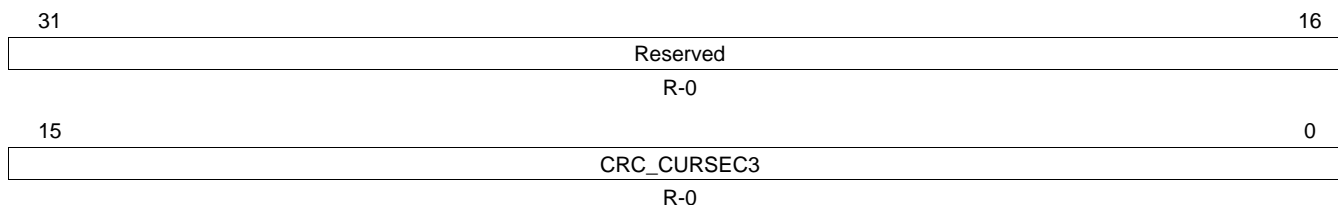
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-40. CRC Sector Counter Preload Register 3 (CRC\_SCOUNT\_REG3) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	CRC_SEC_COUNT3		<b>Channel 3 Sector Counter Preload Register.</b> This register contains the number of sectors in one block of memory.

### 14.4.37 CRC Current Sector Register 3 (CRC\_CURSEC\_REG3)

Figure 14-46. CRC Current Sector Register 3 (CRC\_CURSEC\_REG3) [offset = C8h]



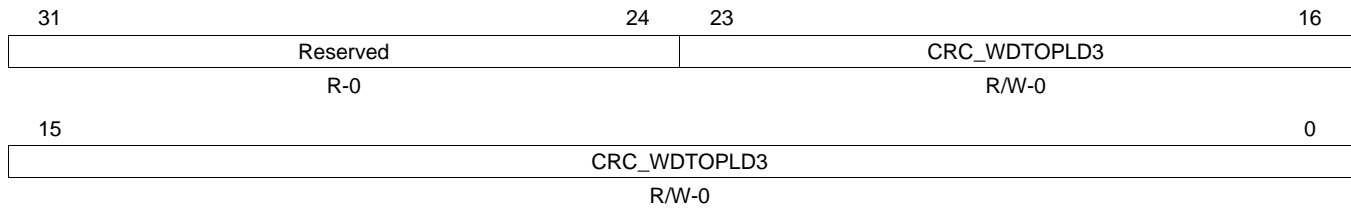
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 14-41. CRC Current Sector Register 3 (CRC\_CURSEC\_REG3) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	CRC_CURSEC3		<b>Channel 3 Current Sector ID Register.</b> In AUTO mode, this register contains the current sector number of which the signature verification fails. The sector counter is a free running up counter. When a sector fails, the erroneous sector number is logged into current sector ID register and the CRC fail interrupt is generated. The sector ID register is frozen until it is read and the CRC fail status bit is cleared by CPU. While it is frozen, it does not capture another erroneous sector number. When this condition happens, an overrun interrupt is generated instead. Once the register is read and the CRC fail interrupt flag is cleared it can capture new erroneous sector number. In Semi-CPU mode, this register is used to indicate the sector number for which the compression complete has last happened.

### 14.4.38 CRC Channel 3 Watchdog Timeout Preload Register A (CRC\_WDTPLD3)

**Figure 14-47. CRC Channel 3 Watchdog Timeout Preload Register A (CRC\_WDTPLD3) [offset = CCh]**



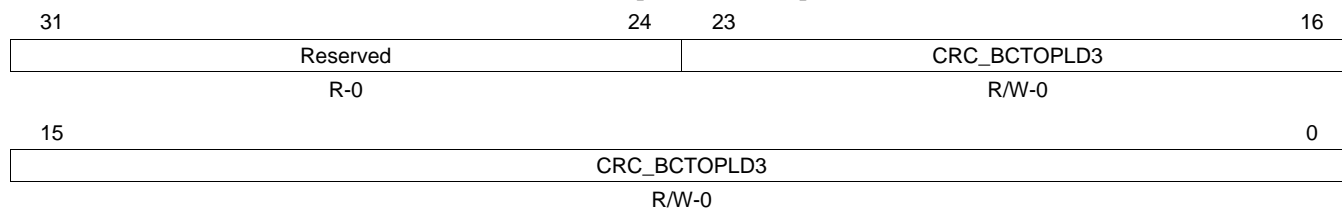
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-42. CRC Channel 3 Watchdog Timeout Preload Register A (CRC\_WDTPLD3) Field Descriptions**

Bit	Field	Value	Description
31-24	Reserved	0	Read returns 0. Writes have no effect.
23-0	CRC_WDTPLD3		<b>Channel 3 Watchdog Timeout Counter Preload Register.</b> This register contains the number of clock cycles within which the DMA must transfer the next block of data patterns. In Semi-CPU mode, this register is used to indicate the sector number for which the compression complete has last happened.

### 14.4.39 CRC Channel 3 Block Complete Timeout Preload Register B (CRC\_BCTOPLD3)

**Figure 14-48. CRC Channel 3 Block Complete Timeout Preload Register B (CRC\_BCTOPLD3)  
[offset = D0h]**



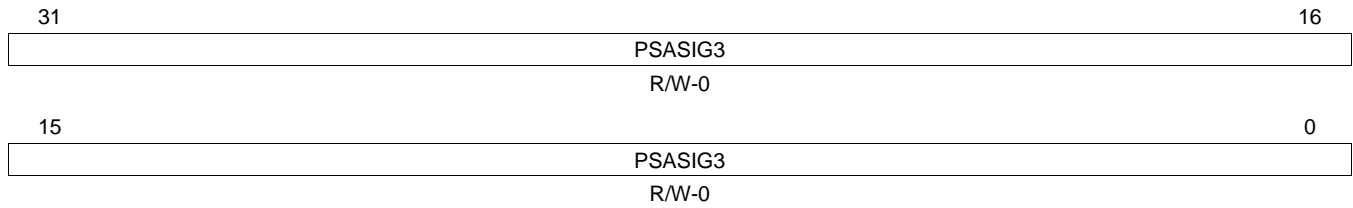
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-43. CRC Channel 3 Block Complete Timeout Preload Register B (CRC\_BCTOPLD3) Field Descriptions**

Bit	Field	Value	Description
31-24	Reserved	0	Read returns 0. Writes have no effect.
23-0	CRC_BCTOPLD3		<b>Channel 3 Block Complete Timeout Counter Preload Register.</b> This register contains the number of clock cycles within which the CRC for an entire block needs to complete before a timeout interrupt is generated.

**14.4.40 Channel 3 PSA Signature Low Register (PSA\_SIGREGL3)**

**Figure 14-49. Channel 3 PSA Signature Low Register (PSA\_SIGREGL3) [offset = E0h]**



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

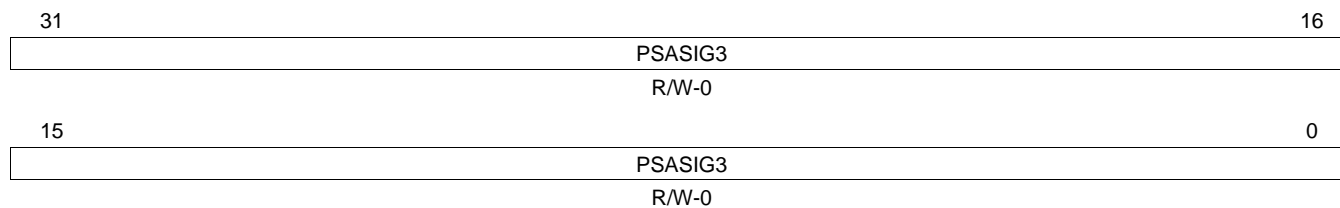
**Table 14-44. Channel 3 PSA Signature Low Register (PSA\_SIGREGL3) Field Descriptions**

Bit	Field	Value	Description
31-0	PSASIG3		<b>Channel 3 PSA Signature Low Register.</b> This register contains the value stored at PSASIG3[31:0] register.



#### 14.4.41 Channel 3 PSA Signature High Register (PSA\_SIGREGH3)

**Figure 14-50. Channel 3 PSA Signature High Register (PSA\_SIGREGH3) [offset = E4h]**



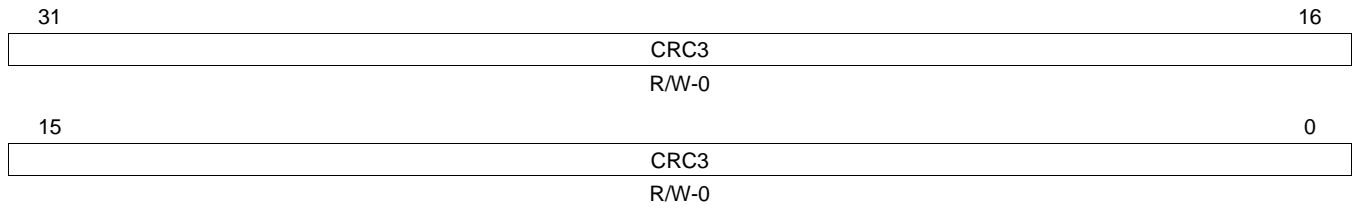
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-45. Channel 3 PSA Signature High Register (PSA\_SIGREGH3) Field Descriptions**

Bit	Field	Value	Description
31-0	PSASIG3		<b>Channel 3 PSA Signature High Register.</b> This register contains the value stored at PSASIG3[63:32] register.

**14.4.42 Channel 3 CRC Value Low Register (CRC\_REGL3)**

**Figure 14-51. Channel 3 CRC Value Low Register (CRC\_REGL3) [offset = E8h]**



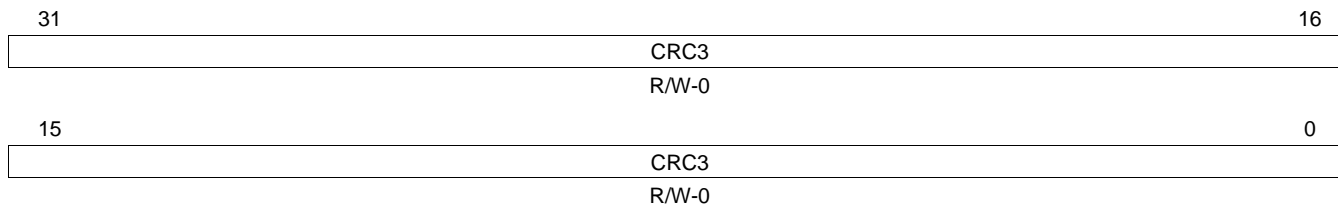
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-46. Channel 3 CRC Value Low Register (CRC\_REGL3) Field Descriptions**

Bit	Field	Value	Description
31-0	CRC3		<b>Channel 3 CRC Value Low Register.</b> This register contains the current known good signature value stored at CRC3[31:0] register.

### 14.4.43 Channel 3 CRC Value High Register (CRC\_REGH3)

Figure 14-52. Channel 3 CRC Value High Register (CRC\_REGH3) [offset = ECh]



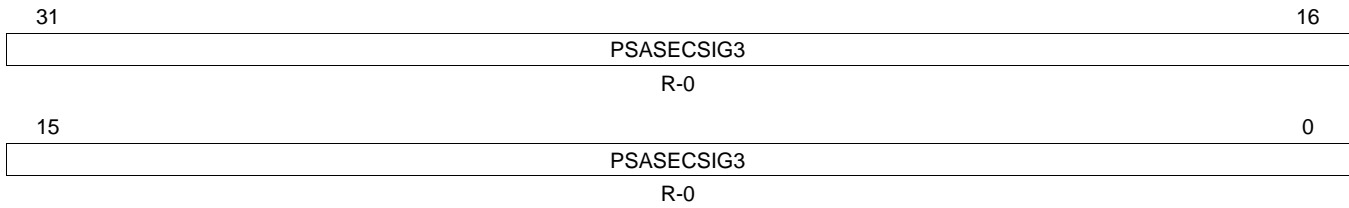
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 14-47. Channel 3 CRC Value High Register (CRC\_REGH3) Field Descriptions

Bit	Field	Value	Description
31-0	CRC3		<b>Channel 3 CRC Value High Register.</b> This register contains the current known good signature value stored at CRC3[63:32] register.

#### 14.4.44 Channel 3 PSA Sector Signature Low Register (PSA\_SECSIGREGL3)

**Figure 14-53. Channel 3 PSA Sector Signature Low Register (PSA\_SECSIGREGL3) [offset = F0h]**



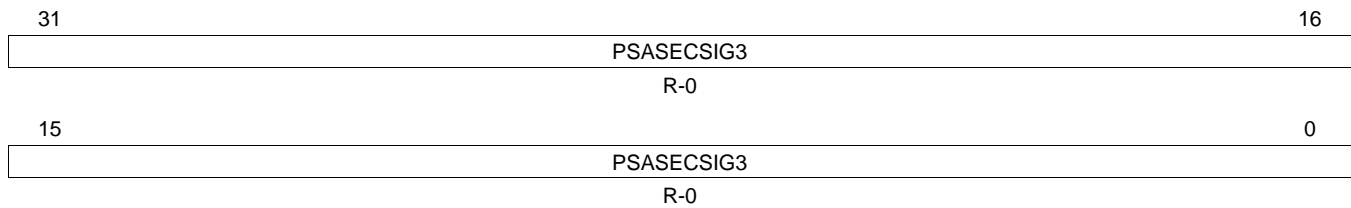
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-48. Channel 3 PSA Sector Signature Low Register (PSA\_SECSIGREGL3) Field Descriptions**

Bit	Field	Value	Description
31-0	PSASECSIG3		<b>Channel 3 PSA Sector Signature Low Register.</b> This register contains the value stored at PSASECSIG3[31:0] register.

### 14.4.45 Channel 3 PSA Sector Signature High Register (PSA\_SECSIGREGH3)

**Figure 14-54. Channel 3 PSA Sector Signature High Register (PSA\_SECSIGREGH3) [offset = F4h]**



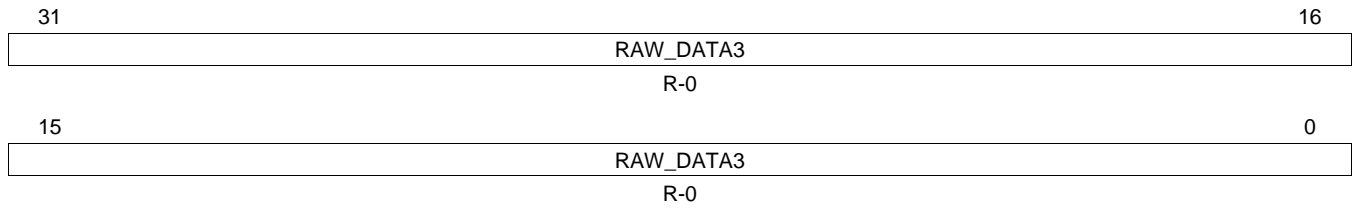
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-49. Channel 3 PSA Sector Signature High Register (PSA\_SECSIGREGH3) Field Descriptions**

Bit	Field	Value	Description
31-0	PSASECSIG3		<b>Channel 3 PSA Sector Signature High Register.</b> This register contains the value stored at PSASECSIG3[63:32] register.

**14.4.46 Channel 3 Raw Data Low Register (RAW\_DATAREGL3)**

**Figure 14-55. Channel 3 Raw Data Low Register (RAW\_DATAREGL3) [offset = F8h]**



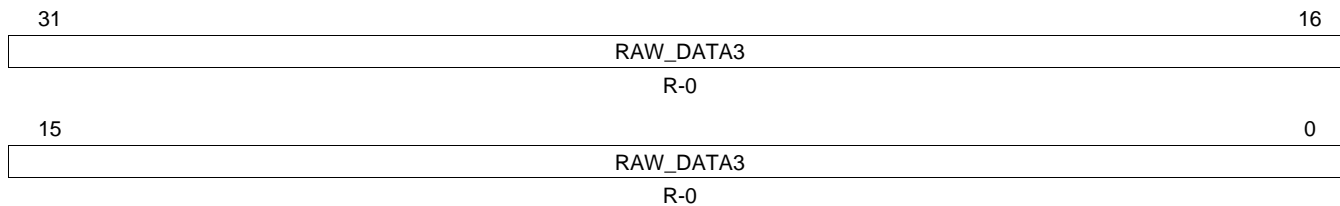
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-50. Channel 3 Raw Data Low Register (RAW\_DATAREGL3) Field Descriptions**

Bit	Field	Value	Description
31-0	RAW_DATA3		<b>Channel 3 Raw Data Low Register.</b> This register contains bit 31:0 of the un-compressed raw data..

### 14.4.47 Channel 3 Raw Data High Register (RAW\_DATAREGH3)

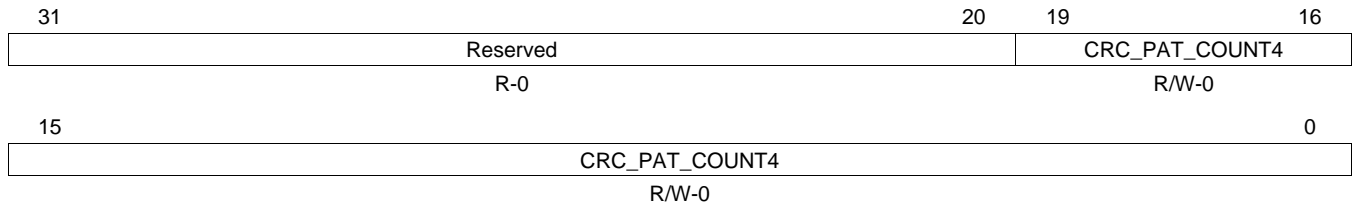
Figure 14-56. Channel 3 Raw Data High Register (RAW\_DATAREGH3) [offset = FCh]



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

Table 14-51. Channel 3 Raw Data High Register (RAW\_DATAREGH3) Field Descriptions

Bit	Field	Value	Description
31-0	RAW_DATA3		<b>Channel 3 Raw Data High Register.</b> This register contains bit 63:32 of the un-compressed raw data..

**14.4.48 CRC Pattern Counter Preload Register 4 (CRC\_PCOUNT\_REG4)**
**Figure 14-57. CRC Pattern Counter Preload Register 4 (CRC\_PCOUNT\_REG4) [offset = 100h]**


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

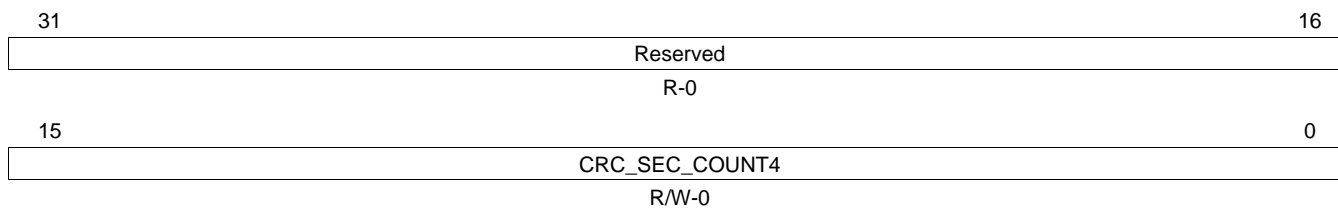
**Table 14-52. CRC Pattern Counter Preload Register 4 (CRC\_PCOUNT\_REG4) Field Descriptions**

Bit	Field	Value	Description
31-20	Reserved	0	Read returns 0. Writes have no effect.
19-0	CRC_PAT_COUNT4		<b>Channel 4 Pattern Counter Preload Register.</b> This register contains the number of data patterns in one sector to be compressed before a CRC is performed.



#### 14.4.49 CRC Sector Counter Preload Register 4 (CRC\_SCOUNT\_REG4)

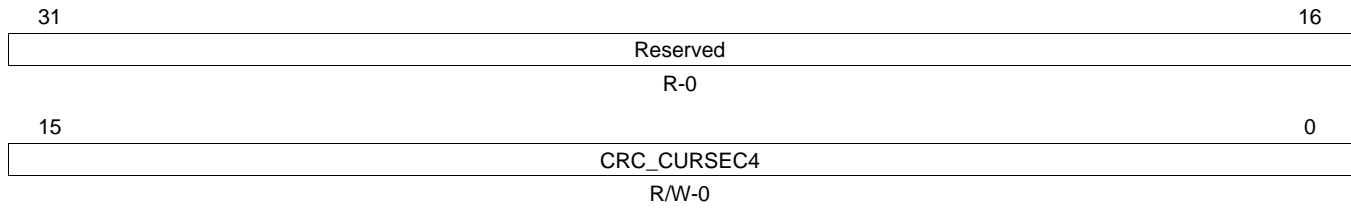
Figure 14-58. CRC Sector Counter Preload Register 4 (CRC\_SCOUNT\_REG4) [offset = 104h]



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

Table 14-53. CRC Sector Counter Preload Register 4 (CRC\_SCOUNT\_REG4) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	CRC_SEC_COUNT4		<b>Channel 4 Sector Counter Preload Register.</b> This register contains the number of sectors in one block of memory.

**14.4.50 CRC Current Sector Register 4 (CRC\_CURSEC\_REG4)**
**Figure 14-59. CRC Current Sector Register 4 (CRC\_CURSEC\_REG4) [offset = 108h]**


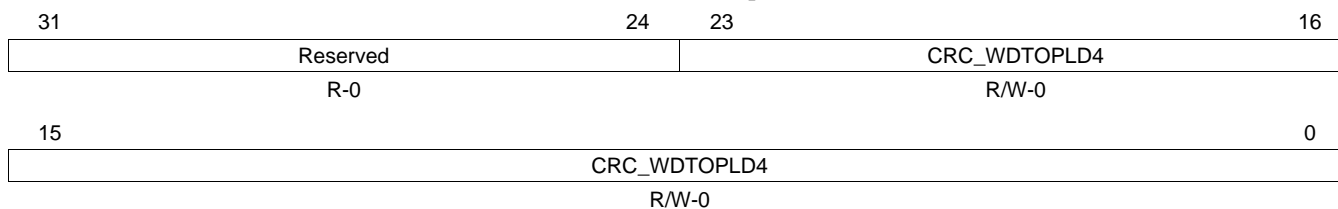
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-54. CRC Current Sector Register 4 (CRC\_CURSEC\_REG4) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	CRC_CURSEC4		<b>Channel 4 Current Sector ID Register.</b> In AUTO mode, this register contains the current sector number of which the signature verification fails. The sector counter is a free running up counter. When a sector fails, the erroneous sector number is logged into current sector ID register and the CRC fail interrupt is generated. The sector ID register is frozen until it is read and the CRC fail status bit is cleared by CPU. While it is frozen, it does not capture another erroneous sector number. When this condition happens, an overrun interrupt is generated instead. Once the register is read and the CRC fail interrupt flag is cleared it can capture new erroneous sector number. In Semi-CPU mode, this register is used to indicate the sector number for which the compression complete has last happened.

### 14.4.51 CRC Channel 4 Watchdog Timeout Preload Register A (CRC\_WDTPLD4)

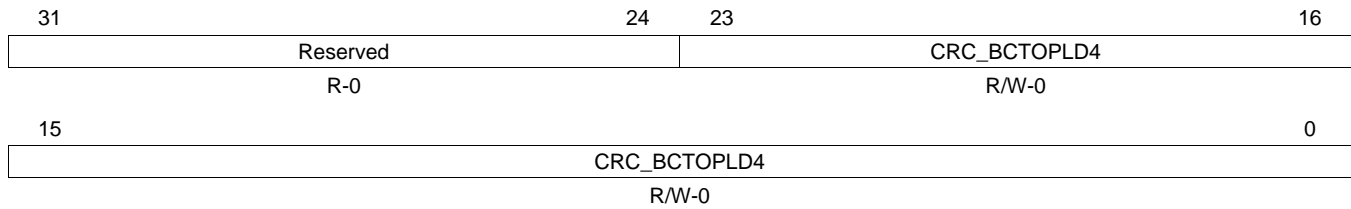
**Figure 14-60. CRC Channel 4 Watchdog Timeout Preload Register A (CRC\_WDTPLD4) [offset = 10Ch]**



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

**Table 14-55. CRC Channel 4 Watchdog Timeout Preload Register A (CRC\_WDTPLD4) Field Descriptions**

Bit	Field	Value	Description
31-24	Reserved	0	Read returns 0. Writes have no effect.
23-0	CRC_WDTPLD4		<b>Channel 4 Watchdog Timeout Counter Preload Register.</b> This register contains the number of clock cycles within which the DMA must transfer the next block of data patterns. In Semi-CPU mode, this register is used to indicate the sector number for which the compression complete has last happened.

**14.4.52 CRC Channel 4 Block Complete Timeout Preload Register B (CRC\_BCTOPLD4)**
**Figure 14-61. CRC Channel 4 Block Complete Timeout Preload Register B (CRC\_BCTOPLD4)  
[offset = 110h]**


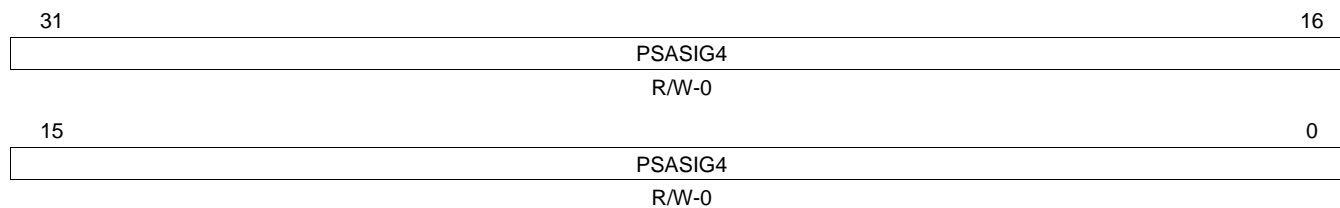
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-56. CRC Channel 4 Block Complete Timeout Preload Register B (CRC\_BCTOPLD4) Field Descriptions**

Bit	Field	Value	Description
31-24	Reserved	0	Read returns 0. Writes have no effect.
23-0	CRC_BOTCPD4		<b>Channel 4 Block Complete Timeout Counter Preload Register.</b> This register contains the number of clock cycles within which the CRC for an entire block needs to complete before a timeout interrupt is generated.

### 14.4.53 Channel 4 PSA Signature Low Register (PSA\_SIGREGL4)

Figure 14-62. Channel 4 PSA Signature Low Register (PSA\_SIGREGL4) [offset = 120h]



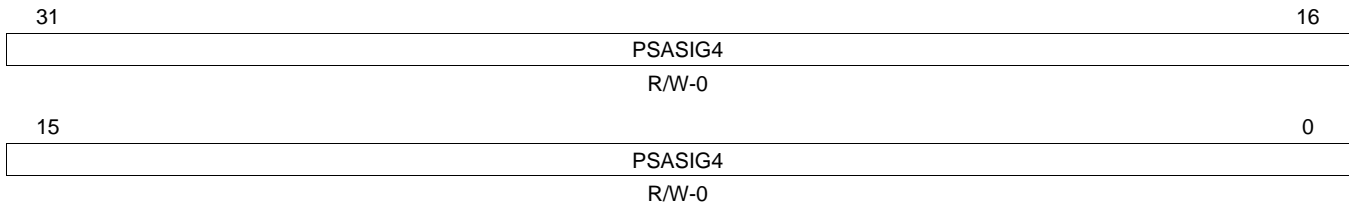
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 14-57. Channel 4 PSA Signature Low Register (PSA\_SIGREGL4) Field Descriptions

Bit	Field	Value	Description
31-0	PSASIG4		<b>Channel 4 PSA Signature Low Register.</b> This register contains the value stored at PSASIG4[31:0] register.

### 14.4.54 Channel 4 PSA Signature High Register (PSA\_SIGREGH4)

**Figure 14-63. Channel 4 PSA Signature High Register (PSA\_SIGREGH4) [offset = 124h]**



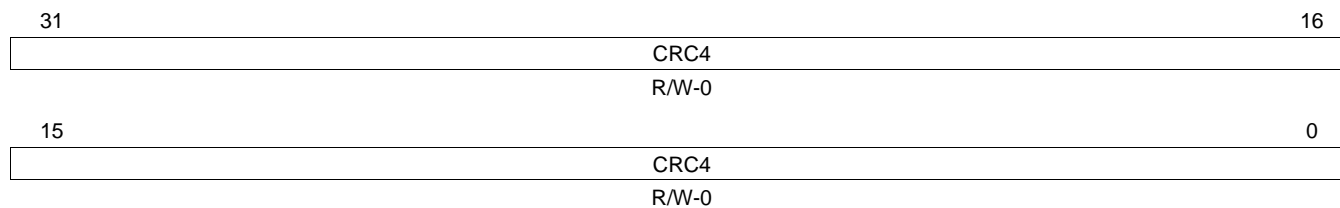
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-58. Channel 4 PSA Signature High Register (PSA\_SIGREGH4) Field Descriptions**

Bit	Field	Value	Description
31-0	PSASIG4		<b>Channel 4 PSA Signature High Register.</b> This register contains the value stored at PSASIG4[63:32] register.

### 14.4.55 Channel 4 CRC Value Low Register (CRC\_REGL4)

**Figure 14-64. Channel 4 CRC Value Low Register (CRC\_REGL4) [offset = 128h]**



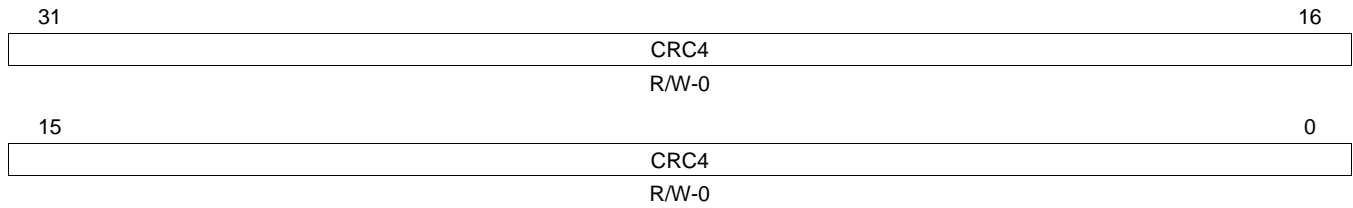
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-59. Channel 4 CRC Value Low Register (CRC\_REGL4) Field Descriptions**

Bit	Field	Value	Description
31-0	CRC4		<b>Channel 4 CRC Value Low Register.</b> This register contains the current known good signature value stored at CRC4[31:0] register.

**14.4.56 Channel 4 CRC Value High Register (CRC\_REGH4)**

**Figure 14-65. Channel 4 CRC Value High Register (CRC\_REGH4) [offset = 12Ch]**



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

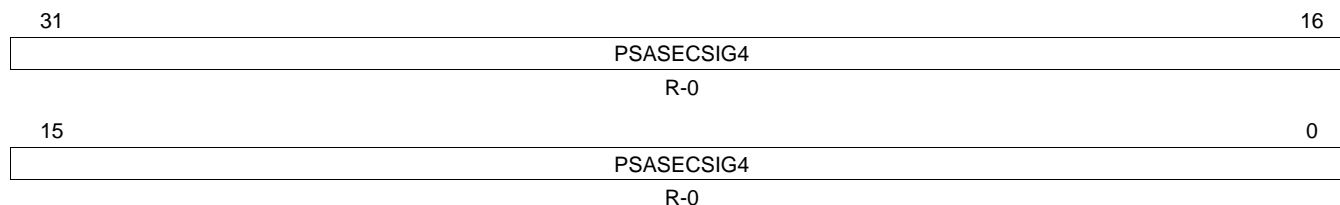
**Table 14-60. Channel 4 CRC Value High Register (CRC\_REGH4) Field Descriptions**

Bit	Field	Value	Description
31-0	CRC4		<b>Channel 4 CRC Value High Register.</b> This register contains the current known good signature value stored at CRC4[63:32] register.



### 14.4.57 Channel 4 PSA Sector Signature Low Register (PSA\_SECSIGREGL4)

**Figure 14-66. Channel 4 PSA Sector Signature Low Register (PSA\_SECSIGREGL4) [offset = 130h]**



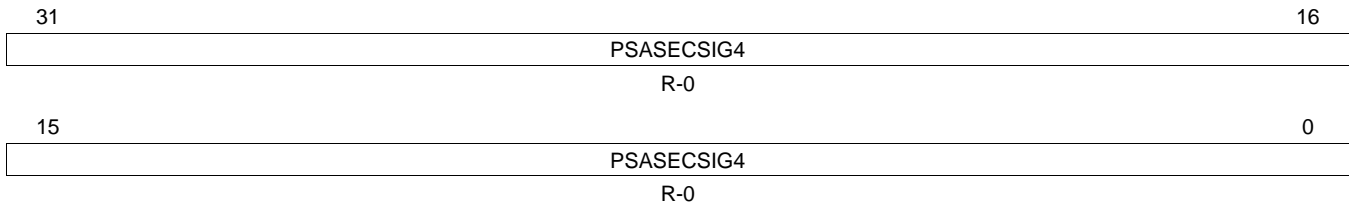
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-61. Channel 4 PSA Sector Signature Low Register (PSA\_SECSIGREGL4) Field Descriptions**

Bit	Field	Value	Description
31-0	PSASECSIG4		<b>Channel 4 PSA Sector Signature Low Register.</b> This register contains the value stored at PSASECSIG4[31:0] register.

### 14.4.58 Channel 4 PSA Sector Signature High Register (PSA\_SECSIGREGH4)

**Figure 14-67. Channel 4 PSA Sector Signature High Register (PSA\_SECSIGREGH4) [offset = 134h]**



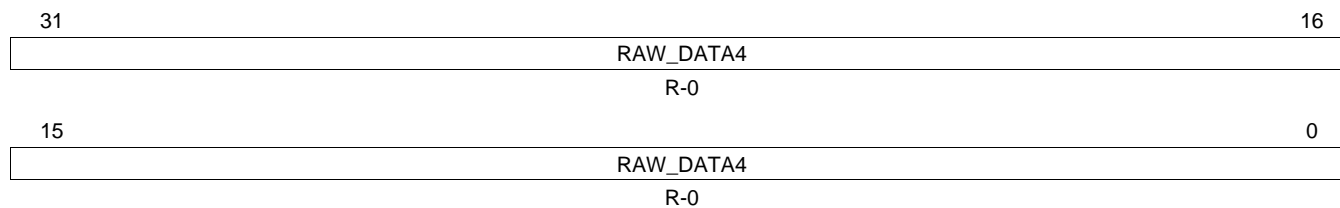
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-62. Channel 4 PSA Sector Signature High Register (PSA\_SECSIGREGH4) Field Descriptions**

Bit	Field	Value	Description
31-0	PSASECSIG4		<b>Channel 4 PSA Sector Signature High Register.</b> This register contains the value stored at PSASECSIG4[63:32] register.

### 14.4.59 Channel 4 Raw Data Low Register (RAW\_DATAREGL4)

**Figure 14-68. Channel 4 Raw Data Low Register (RAW\_DATAREGL4) [offset = 138h]**



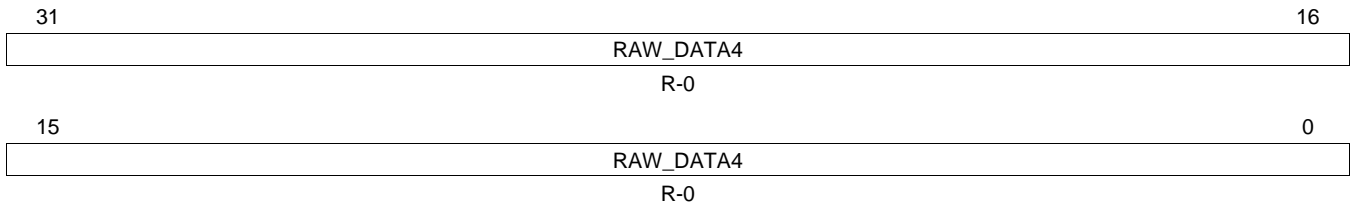
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-63. Channel 4 Raw Data Low Register (RAW\_DATAREGL4) Field Descriptions**

Bit	Field	Value	Description
31-0	RAW_DATA4		<b>Channel 4 Raw Data Low Register.</b> This register contains bit 31:0 of the un-compressed raw data..

**14.4.60 Channel 4 Raw Data High Register (RAW\_DATAREGH4)**

**Figure 14-69. Channel 4 Raw Data High Register (RAW\_DATAREGH4) [offset = 13Ch]**



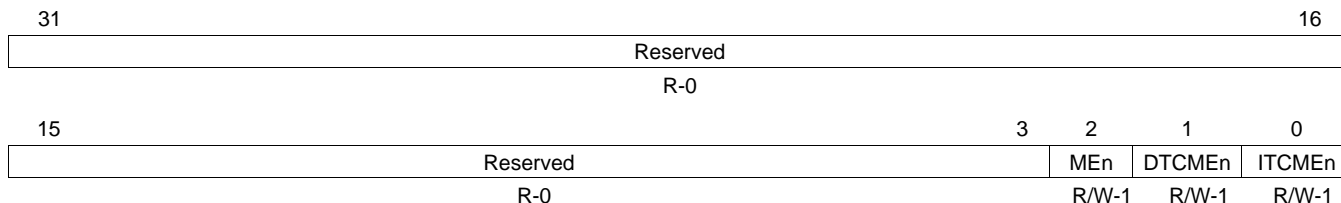
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-64. Channel 4 Raw Data High Register (RAW\_DATAREGH4) Field Descriptions**

Bit	Field	Value	Description
31-0	RAW_DATA4		<b>Channel 4 Raw Data High Register.</b> This register contains bit 63:32 of the un-compressed raw data.

### 14.4.61 Data Bus Selection Register (MCRC\_TRACE\_BUS\_SEL)

Figure 14-70. Data Bus Selection Register (MCRC\_TRACE\_BUS\_SEL) [offset = 140h]



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

Table 14-65. Channel 4 Data Bus Selection Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Read returns 0. Writes have no effect.
2	MEn	0	Enable/disables the tracing of Peripheral Bus Master
		0	Tracing of Peripheral Bus Master has been disabled
		1	Tracing of Peripheral Bus Master master bus has been enabled
1	DTCMEn	0	Enable/disables the tracing of data TCM
		0	Tracing of System Odd and Even RAM buses have been disabled
		1	Tracing of System Odd and Even RAM buses have been enabled
0	ITCMEn	0	Enable/disables the tracing of instruction TCM
		0	Tracing of Flash data bus has been disabled
		1	Tracing of Flash data bus has been enabled

## ***Vectored Interrupt Manager (VIM) Module***

---



---



---

This chapter describes the behavior of the vectored interrupt manager (VIM) module of the device family.

Topic	Page
<b>15.1 Overview</b> .....	<b>604</b>
<b>15.2 Device Level Interrupt Management</b> .....	<b>605</b>
<b>15.3 Interrupt Handling Inside VIM</b> .....	<b>607</b>
<b>15.4 Interrupt Vector Table (VIM RAM)</b> .....	<b>612</b>
<b>15.5 VIM Wakeup Interrupt</b> .....	<b>615</b>
<b>15.6 Capture Event Sources</b> .....	<b>616</b>
<b>15.7 Examples</b> .....	<b>617</b>
<b>15.8 Registers</b> .....	<b>619</b>

## 15.1 Overview

The vectored interrupt manager (VIM) provides hardware assistance for prioritizing and controlling the many interrupt sources present on a device. Interrupts are caused by events outside of the normal flow of program execution. Normally, these events require a timely response from the central processing unit (CPU); therefore, when an interrupt occurs, the CPU switches execution from the normal program flow to an interrupt service routine (ISR).

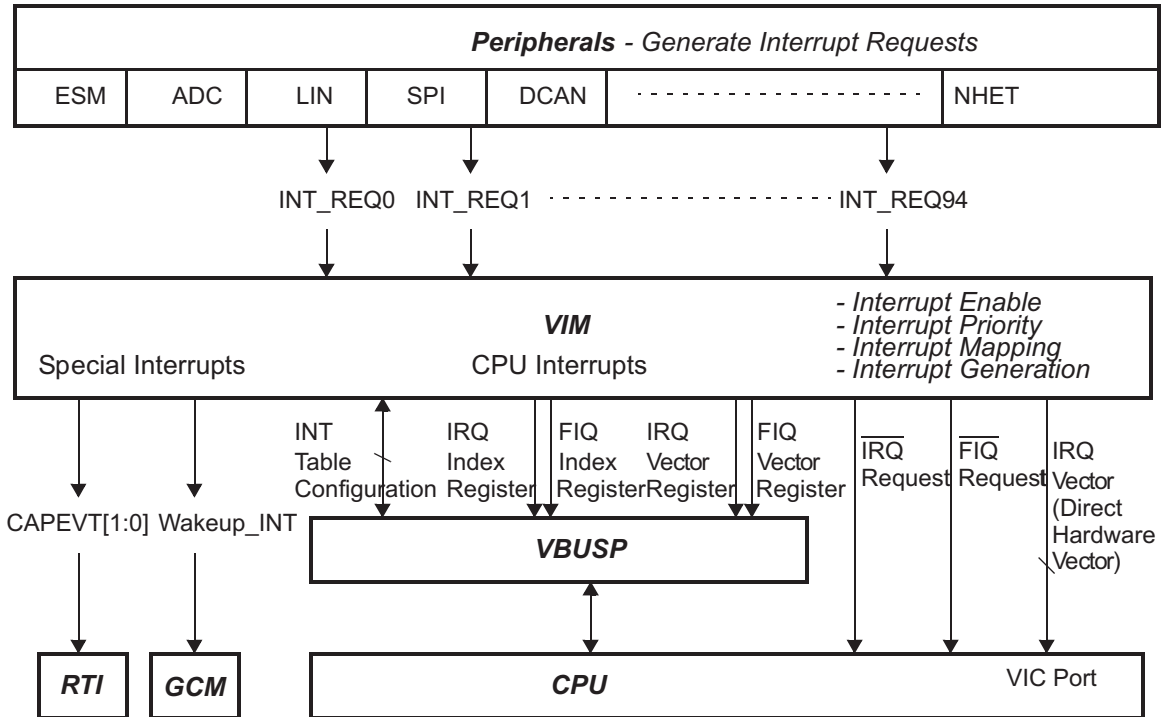
The VIM module has the following features:

- Supports 95 interrupt channels, in both register vectored interrupt and hardware vectored interrupt mode.
  - Provides IRQ vector directly to the CPU VIC port.
  - Provides FIQ/IRQ vector through registers.
  - Provides programmable priority and enable for interrupt request lines.
- Provides a direct hardware dispatch mechanism for fastest IRQ dispatch.
- Provides two software dispatch mechanisms for backward compatibility with earlier generation of TI processors.
  - Index interrupt
  - Register vectored interrupt
- Parity protected vector interrupt table against soft errors.

## 15.2 Device Level Interrupt Management

A block diagram of device level interrupt handling is shown in Figure 15-1. When an event occurs within a peripheral, the peripheral makes an interrupt request to the VIM. Then, VIM Prioritizes the requests from peripherals and provides the address of the highest interrupt service routine (ISR) to the CPU. Finally, CPU dispatch instructions in the ISR. Section 15.2.1 through Section 15.2.3 provide additional details about these three steps.

Figure 15-1. Device Level Interrupt Block Diagram



### 15.2.1 Interrupt Generation at the Peripheral

Interrupt generation begins when an event occurs within a peripheral module. Some examples of interrupt-capable events are expiration of a counter within a timer module, receipt of a character in a communications module, and completion of a conversion in an analog-to-digital converter (ADC) module. Some device peripherals are capable of requesting interrupts on more than one interrupt request line.

Interrupts are not always generated when an event occurs; the peripheral must make an interrupt request to the VIM based on the event occurrence. Typically, the peripheral contains:

- An interrupt flag bit for each event to signify the event occurrence.
- An interrupt enable bit to control whether the event occurrence causes an interrupt request to the VIM.

### 15.2.2 Interrupt Handling at the CPU

The ARM CPU provides two vectors for interrupt requests—fast interrupt requests (FIQs) and normal interrupt requests (IRQs). FIQs are higher priority than IRQs, and FIQ interrupts may interrupt IRQ interrupts.

**NOTE:** The FIQ implemented in Cortex-R4F is Non-Maskable Fast Interrupts (NMFI). Once FIQ is enabled (by clearing F bit in CPSR), it can NOT be disabled by setting F bit in CPSR. Only a reset or an FIQ will be able to set the F bit in CPSR. By hardware, Non Maskable FIQ are not reentrant.



After reset (power reset or warm reset), both FIQ and IRQ are disabled. The CPU may enable these interrupt request channels individually within the CPSR (Current Program Status Register); CPSR bits 6 and 7 must be cleared to enable the FIQ (bit 6) and IRQ (bit 7) interrupt requests at the CPU. CPSR is writable in privilege mode only. [Example 15-2](#) shows how to enable the IRQ and FIQ through CPSR.

When the CPU receives an interrupt request, the CPSR mode field changes to either FIQ or IRQ mode. When an IRQ interrupt is received, the CPU disables other IRQ interrupts by setting CPSR bit 7. When an FIQ interrupt is received, the CPU disables both IRQ and FIQ interrupts by setting CPSR bits 6 and 7.

A write of 1 to CPSR bit 7 disables the IRQ from CPU. However, a write of 1 to CPSR bit 6 leaves it unchanged. [Example 15-2](#) also shows how to disable the IRQ through CPSR.

### 15.2.3 Software Interrupt Handling Options

The device supports three different possibilities for software to handle interrupts

1. Index interrupts mode (compatible with TMS470R1x legacy code)
 

After the interrupt is received by the CPU, the CPU branches to 0x18 (IRQ) or 0x1C (FIQ) to execute the main ISR. The main ISR routine reads the offset register (IRQINDEX, FIQINDEX) to determine the source of the interrupt.

This mode is compatible with the TMS470R1x (CIM) module and provides the same interrupt registers. This mode could be used if legacy code needs to be reused, porting it from the TMS470R1x family. However, software thus imported will not benefit from the whole range of performance the VIM is capable of.

To port legacy software, the interrupt vector at 0x18 (IRQ) or 0x1C (FIQ) only needs to be a branch statement to a software interrupt table. The software interrupt table reads the pending interrupt from a vector offset register (FIQINDEX[7:0] for FIQ interrupts and IRQINDEX[7:0] for IRQ interrupts). All pending interrupts can be viewed in the INTREQ register. [Example 15-4](#) shows how to respond to FIQ with short latency in this mode.
2. Register vectored interrupts (automatically provide vector address to application)
 

Before enabling interrupts, the application software also has to initiate the interrupt vector table (VIM RAM).

Once the VIM receives an interrupt, it loads the address of ISR from interrupt vector table, and store it into the interrupt vector register (IRQVECREG for IRQ interrupt, FIQVECREG for FIQ interrupt).

After the interrupt is received by the CPU, the CPU executes the instruction placed at 0x18 or 0x1C (IRQ or FIQ vector) to load the address of ISR (interrupt vector) from the interrupt vector register. [Example 15-3](#) illustrates the configuration for the exception vectors using this mode.
3. Hardware vectored interrupts (automatically dispatch to ISR, IRQ only)
 

Before enabling interrupts, the application software must initiate the interrupt vector table (VIM RAM) pointing to the ISR for each interrupt channel.

After the interrupt (IRQ) is received by the CPU, CPU reads the address of ISR directly from the interface with VIM (VIC port) instead of branching to 0x18. The CPU will branch directly to the ISR. The hardware vectored interrupt behavior must be explicitly enabled by setting the vector enable (VE) bit in the CP15 R1 register. This bit resets to 0, so that the default state after reset is backward compatible to earlier ARM CPU. [Example 15-1](#) shows how to enable the hardware vectored interrupt.

---

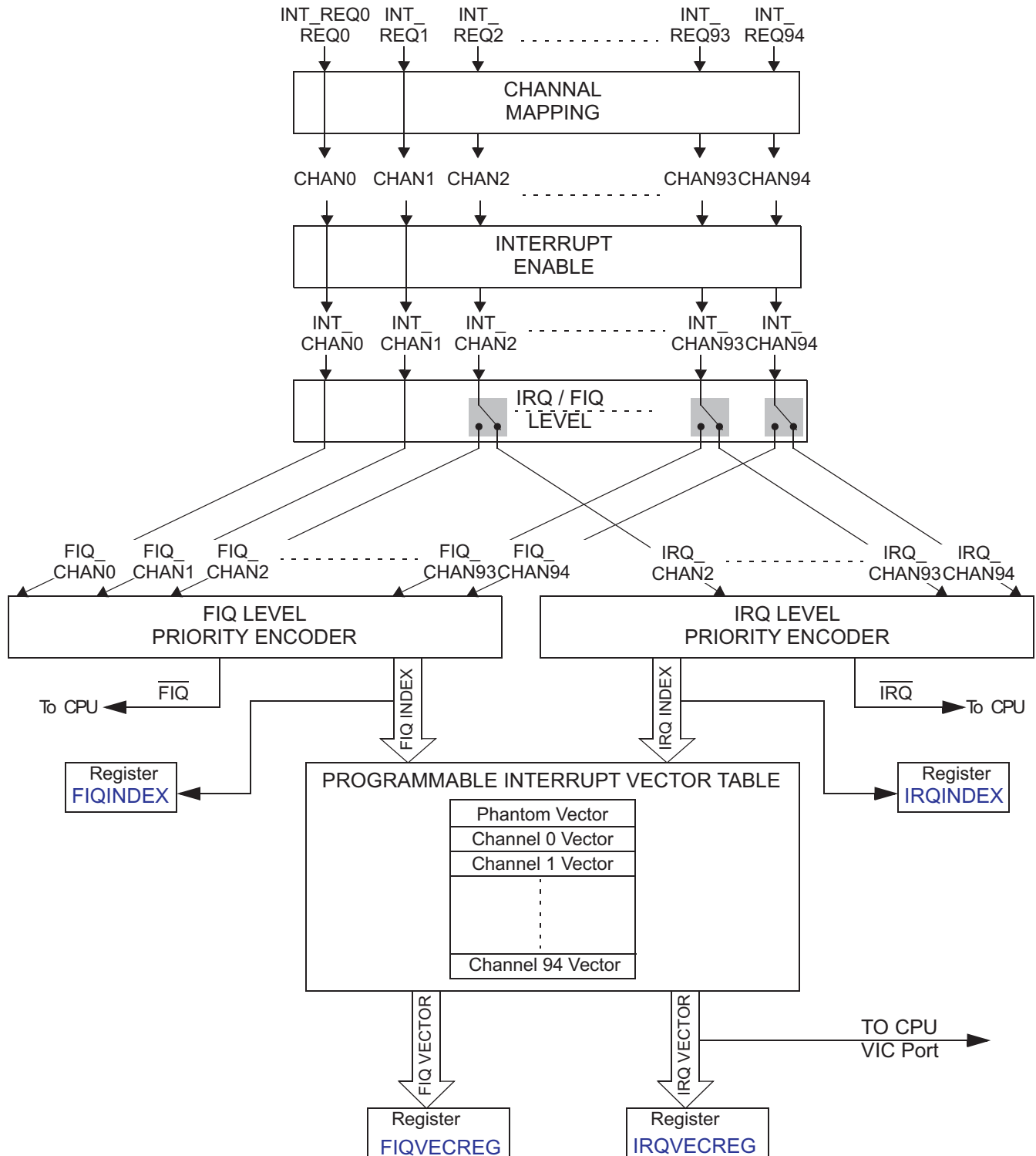
**NOTE:** This mode is NOT available for FIQ.

---

### 15.3 Interrupt Handling Inside VIM

A block diagram of the interrupt handling inside VIM is shown in [Figure 15-2](#)

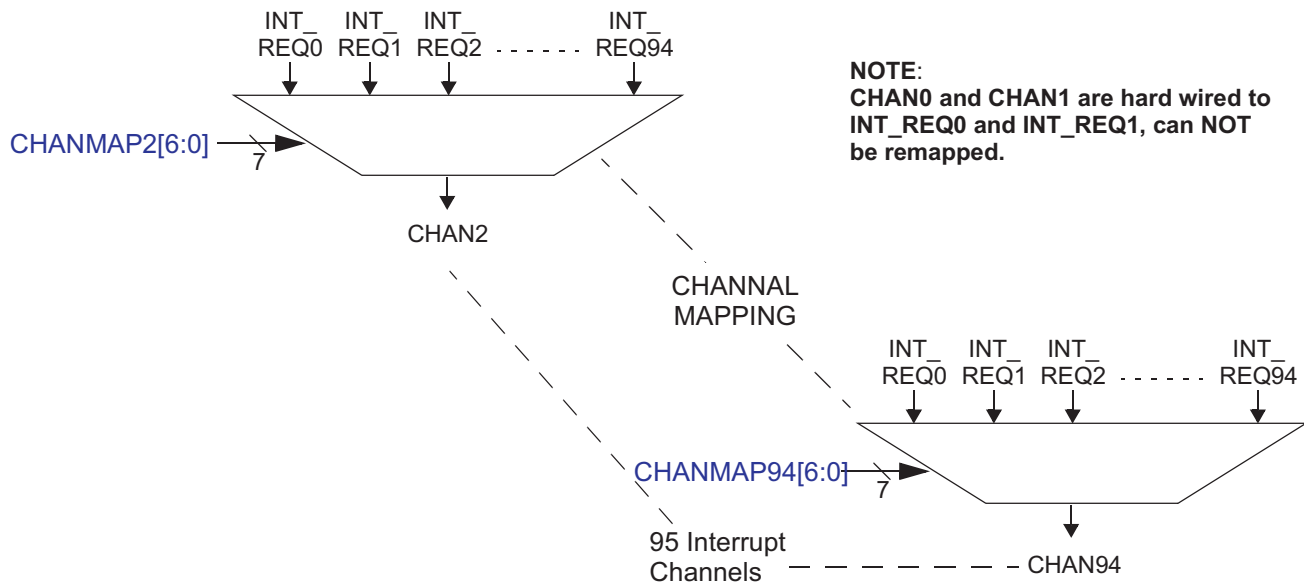
**Figure 15-2. VIM Interrupt Handling Block Diagram**



### 15.3.1 VIM Interrupt Channel Mapping

The VIM support 96 interrupt channels (including phantom interrupt). A block diagram of the VIM interrupt requests arrangement from peripheral modules to the interrupt channels, is provided in Figure 15-3. For each interrupt channel, CHANx, there is a corresponding mapping register bit field CHANMAPx[6:0], which determines which VIM request the interrupt channel maps to. With this scheme, the same request can be mapped to multiple channels. Since a lower numbered channel in each FIQ and IRQ has higher priority. The programmability of the VIM allows software to control the interrupt priority.

**Figure 15-3. VIM Channel Mapping**

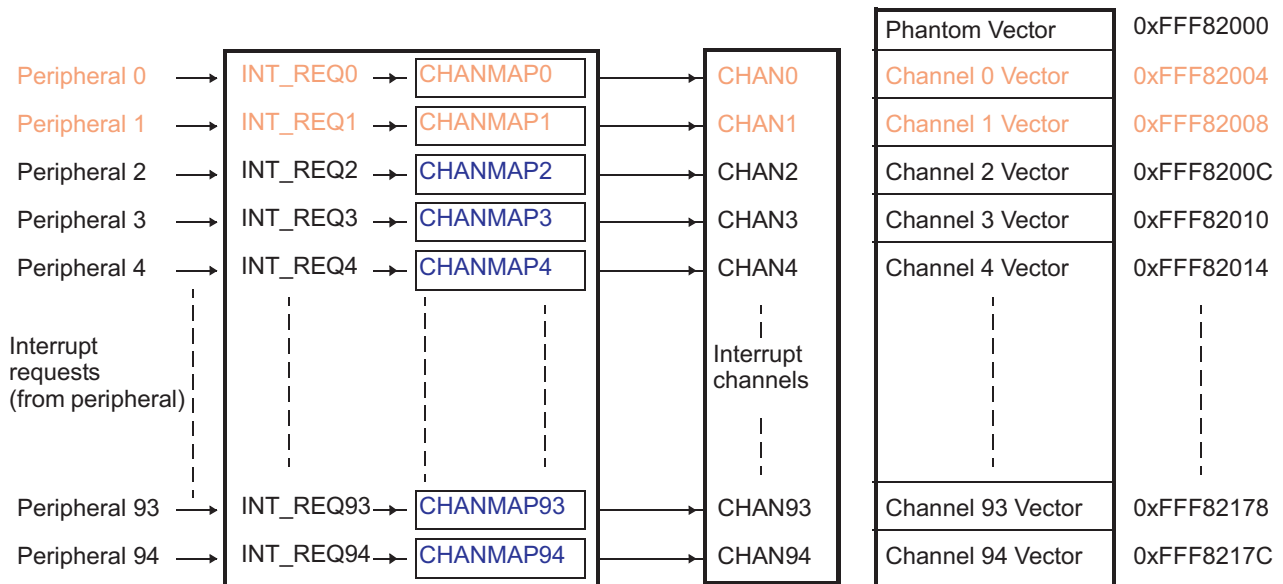


**NOTE: CHAN95**

CHAN95 has no dedicated interrupt vector table entry. Therefore, CHAN95 shall NOT be remapped to other INT\_REQ (INT\_REQ95 is reserved at device level).

In the reset state, the VIM maps all of the interrupt requests in the system to their respective interrupt channels. Figure 15-4 shows the default state following the reset.

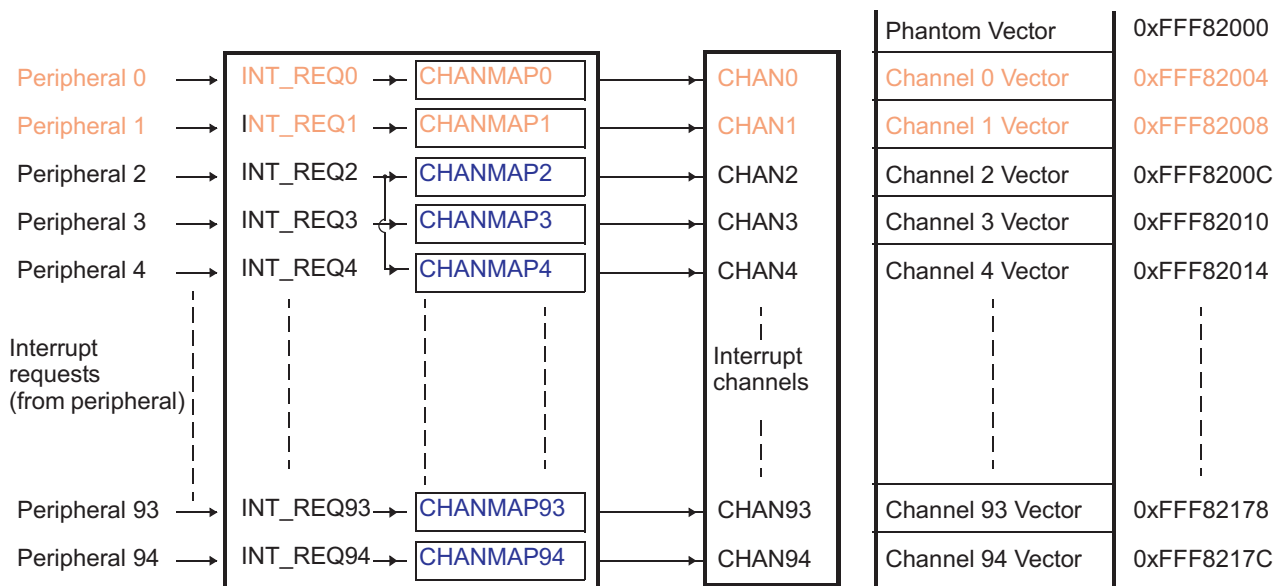
Figure 15-4. VIM in Default State



**NOTE:** CHAN0 and CHAN1 are hard wired to INT\_REQ0 and INT\_REQ1, can NOT be remapped.

Figure 15-5 shows the VIM INT2 is remapped to both Channel 2 and 4, and INT3 is mapped to channel 3.

Figure 15-5. VIM in a Programmed State



**NOTE:** CHAN0 and CHAN1 are hard wired to INT\_REQ0 and INT\_REQ1, can NOT be remapped.

**NOTE:** By mapping INT2 to channel 2 and channel 4, and mapping INT3 to channel 3, it is possible for the software to change the priority dynamically by changing the ENABLE register (REQENASET and REQENACLR). When channel 2 is enabled, the priority is:

1. INT0
2. INT1
3. INT2
4. INT3

Disabling channel 2, the priority becomes:

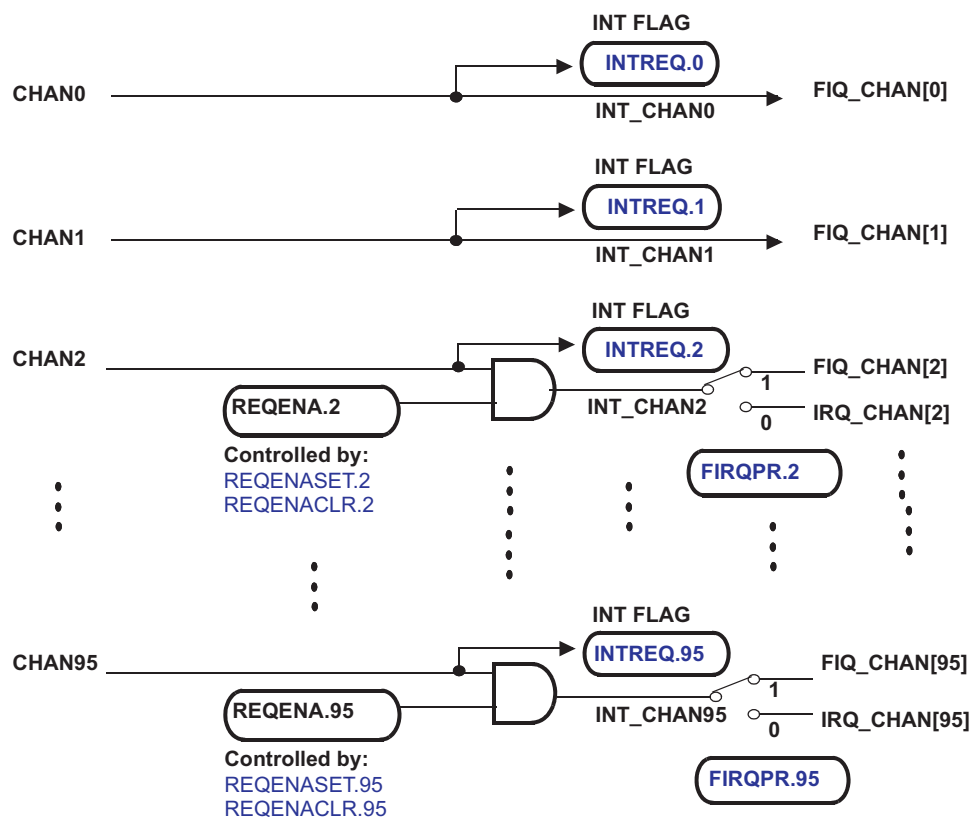
1. INT0
2. INT1
3. INT3
4. INT2

### 15.3.2 VIM Input Channel Management

As shown in Figure 15-6, the VIM enables channels on a channel-by-channel basis (in the REQENASET and REQENACLR registers); unused channels may be masked to prevent spurious interrupts.

**NOTE:** The interrupt ENABLE register does not affect the value of INTREQ.

**Figure 15-6. Interrupt Channel Management**



By default, Interrupt CHAN0 is mapped to ESM (Error Signal Module) high level interrupt and CHAN1 is reserved for other NMI. For safety reasons, these two channels are mapped to FIQ only and can **NOT** be disabled through ENABLE registers.

**NOTE: NMI Channel**

Channel 0 and channel 1 are not maskable by the REQENASET / REQENACLK bit and both channels are routed exclusively to FIQ/NMI request line (FIRQPR0 and FIRQPR1 have no effect).

---

The VIM prioritizes the received interrupts based upon a programmed prioritization scheme. The VIM can send two interrupt requests to the CPU simultaneously—one IRQ and one FIQ. If both interrupt types are enabled at the CPU level, then the FIQ has greater priority and is handled first. Each interrupt channel, except channel 0 and 1, can be assigned to send either an FIQ or IRQ request to the CPU (in the FIRQPR register).

The VIM provides a default prioritization scheme, which sends the lowest numbered active channel (in each FIQ and IRQ classes) to the CPU. Within the FIQ and IRQ classes of interrupts, the lowest channel has the highest priority interrupt. The channel number is programmable through register CHANMAPx.

After the VIM has generated the vector corresponding to the highest active IRQ, it updates the FIQINDEX or the IRQINDEX register, depending on the class of interrupt. Then, it accesses the interrupt vector table using the vector value to fetch the address of the corresponding ISR. If the request is an FIQ class interrupt, the address read from the interrupt vector table, is written to the FIQVECREG register. If the request is an IRQ class interrupt, the address is written to the IRQVECREG register and put on the VIC port of the CPU (in case of hardware vectored interrupt is enabled).

All of the interrupt registers are updated when a new high priority interrupt line becomes active.



---

**NOTE:** The PARFLG register is only for bypassing the interrupt vector table in case of a parity fault. It should be used only to maintain the interrupt vector table bypassed. The checking of the parity fault should be done in the error signalling module (ESM) module where all parity errors are flagged.

---

Since the interrupt vector table may have an error, the FBPARERR register will provide to the VIC port, IRQVECREG and FIQVECREG, a fall-back address to an ISR that can restore the interrupt vector table content. The FBPARERR register should be set before initializing the interrupt in the interrupt vector table, to avoid branching to an unpredictable location.

The normal operation is restored when the PARFLG is cleared by the CPU. It is recommended to restore the content of the VIM before clearing the PARFLG.

The parity error signal is forwarded to the ESM.

### **15.4.2 Enabling and Controlling the VIM Parity**

The polarity of VIM parity is controlled by the DEVCR1 register in the system module (address 0xFFFFFDC). The parity enable is controlled by the PARCTL register. After reset, the parity is disabled.

Parity checking can be enabled by writing 0xA (1010b) in the PARENA[3:0] bit field of the PARCTL register. The default polarity is odd. The polarity can be changed to even by writing 5 (0101b) in the DEVPARSEL[3:0] bit field of the DEVCR1 register.

### **15.4.3 Interrupt Vector Table Initialization**

After reset, the interrupt vector table content, including the parity bits is not initialized. Therefore, the CPU need to initialize all of the interrupt addresses into the table, before enabling the corresponding interrupt channel. If parity is required, this initialization should be done after the parity functionality is enabled. In this way, the corresponding parity bit will be automatically updated. This initialization is only required when vectored interrupts are used, index interrupt management does not need the table to be initialized.

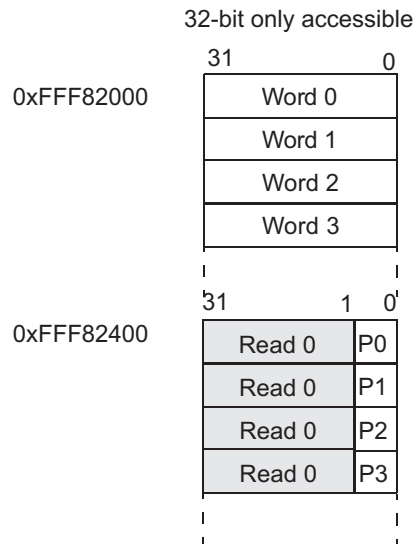
### **15.4.4 Interrupt Vector Table Parity Testing**

To test the parity checking mechanism, the parity RAM allows manual insertion of faults. This option is implemented by the test bit in the PARCTL register. Once the bit is set, the parity bits are mapped to 0xFFFF82400. After that, user can force faults into the parity bits. Finally, the parity error can be triggered by reading interrupt vector table (not parity bit) from VIM or CPU.

The interrupt vector table parity can also be verified by inserting faults into interrupt vector table. Once the VIM parity is disabled in system module, user can modify interrupt vector table without impacting the parity bit. After user re-enable interrupt vector table parity, the parity error can be triggered by reading interrupt vector table from VIM or CPU.



**Figure 15-8. Parity Bit Mapping**

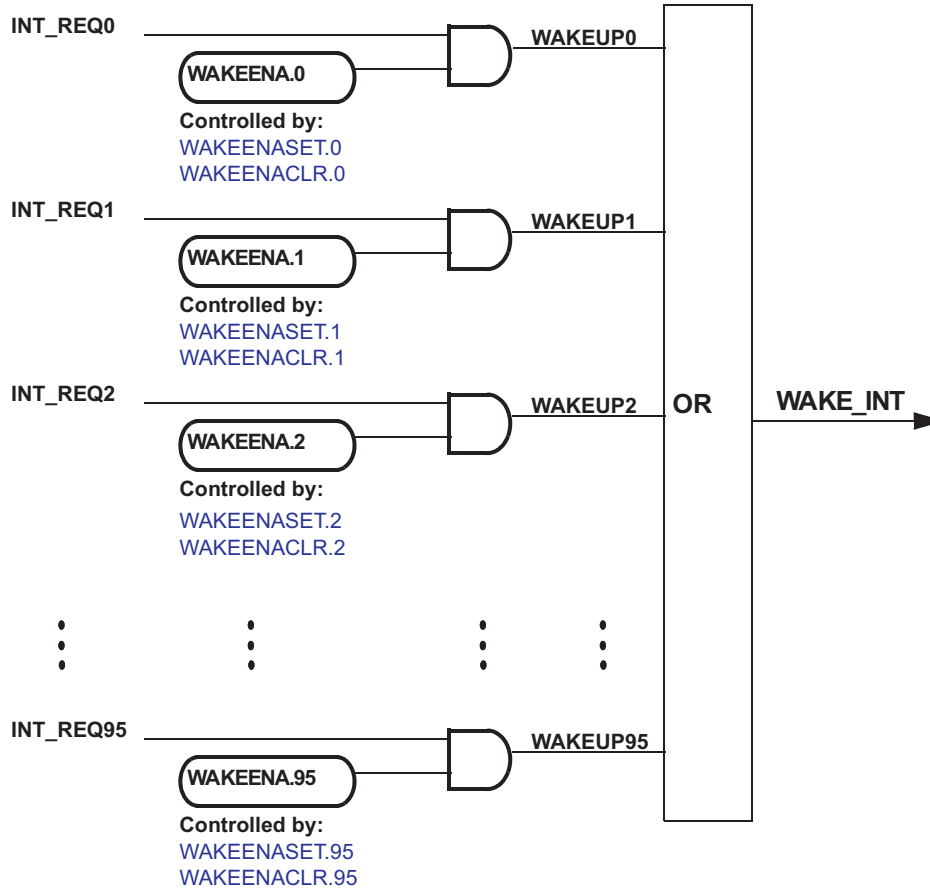


### 15.5 VIM Wakeup Interrupt

The wakeup interrupts are used to come out of low power mode (LPM). Any interrupt requests can be used to wake up the device. After reset, all interrupt requests are set to wake up from LPM. However, the VIM can mask unwanted interrupt lines for wake-up by using the WAKEENASET and WAKEENACLR register. The value in REQENASET / REQENACLR does NOT impact the wakeup interrupt.

As shown in Figure 15-9, the WAKEENASET and WAKEENACLR registers will enable/disable an interrupt for wake-up from lowpower mode. All wake-up interrupts are “ORed” into a single signal WAKE\_INT connected to the Global Clock Module.

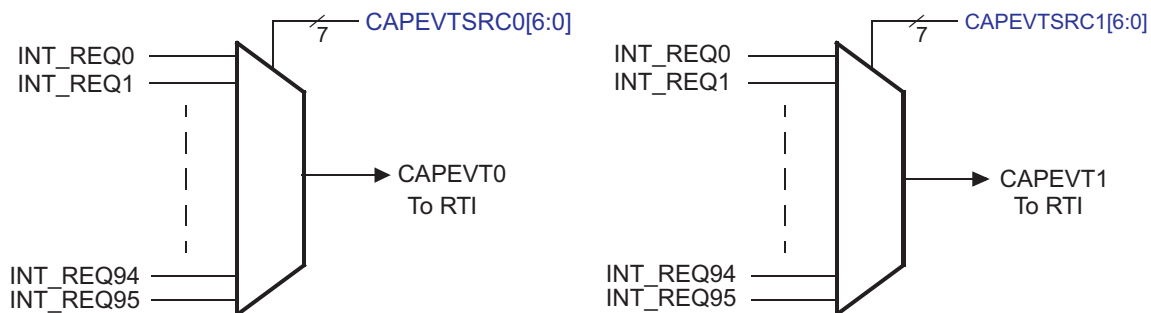
Figure 15-9. Detail of the IRQ Input



## 15.6 Capture Event Sources

The VIM can select any of the 96 interrupt request to generate up to two capture events for the real-time interrupt (RTI) module (see [Figure 15-10](#)). The value in REQENASET / REQENACLR does NOT impact the capture event. Two registers ([Section 15.8.16](#)) are available, one for each capture event source.

**Figure 15-10. Capture Event Sources**



## 15.7 Examples

The following sections provide examples about the operation of the VIM.

### 15.7.1 Examples - Configure CPU To Receive Interrupts

[Example 15-1](#) shows how to set the vector enable (VE) bit in the CP15 R1 register to enable the hardware vector interrupt. [Example 15-2](#) shows how to enable/disable the IRQ and FIQ through CPSR. As a convention, the program who calls these subroutines shall preserve register R1 if needed. [Example 15-2](#) can ONLY run in privileged mode. However, in USER mode, the application software can force the program into software interrupt by instruction 'SWI'. Then, in the software interrupt service routine, user can write register SPSR, which is the copy of CPSR in this exception mode.

#### Example 15-1. Enable Hardware Vector Interrupt (IRQ Only)

```

_HW_Vec_Init
MRC p15 ,#0 ,R1 ,c1 ,c0 ,#0
ORR R1 ,R1 ,#0x01000000      ; Mask 0-31 bits except bit 24 in Sys
                             ; Ctrl Reg of CORTEX-R4
MCR p15 ,#0 ,R1 ,c1 ,c0 ,#0 ; Enable bit 24
MOV PC, LR

```

#### Example 15-2. Enable/Disable IRQ/FIQ through CPSR

```

FIQENABLE .equ 0x40
IRQENABLE .equ 0x80
.....
_Enable_Fiq
MRS R1, CPSR
BIC R1, R1, #FIQENABLE
MSR CPSR, R1
MOV PC, LR
.....
_Disable_Irq
MRS R1, CPSR
ORR R1, R1, #IRQENABLE
MSR CPSR, R1
MOV PC, LR
.....
_Enable_Irq
MRS R1, CPSR
BIC R1, R1, #IRQENABLE
MSR CPSR, R1
MOV PC, LR

```

### 15.7.2 Examples - Register Vector Interrupt and Index Interrupt Handling

[Example 15-3](#) illustrates the configuration for the exception vectors in Register Vector Interrupt handling. After the interrupt is received by the CPU, the CPU branches to 0x18 (IRQ) or 0x1C (FIQ). The instruction placed here should be *LDR PC, [PC,#-0x1B0]*. The pending ISR address is written into the corresponding vector register (IRQVECREG for IRQ, FIQVECREG for FIQ). The CPU reads the content of the register and branches to the ISR.

#### Example 15-3. Exception Vector Configuration for VIM Vector

```

.sect ".intvecs"
00000000h b _RESET          ; RESET interrupt
00000004h b _UNDEF_INST_INT ; UNDEFINED INSTRUCTION interrupt
00000008h b _SW_INT        ; SOFTWARE interrupt

```

**Example 15-3. Exception Vector Configuration for VIM Vector (continued)**

```

0000000Ch b _ABORT_PREF_INT      ; ABORT (PREFETCH) interrupt
00000010h b _ABORT_DATA_INT      ; ABORT (DATA) interrupt
00000014h b #-8                  ; Reserved
00000018h ldr pc,[pc,#-0x1B0]    ; IRQ interrupt
0000001Ch ldr pc,[pc,#-0x1B0]    ; FIQ interrupt

```

**NOTE:** Program Counter (PC) always pointers two instructions beyond the current executed instruction. In this case, PC equals to '0x18 or 0x1C + 0x08'. The *LDR* instruction load the memory at 'PC - 0x1B0', which is '0x18 or 0x1C + 0x08 - 0x1B0 = 0xFFFFFE70 or 0xFFFFFE74'. These are the address of IRQVECREG and FIQVECREG, which store the pending ISR address.

[Example 15-4](#) shows a fast response to the FIQ interrupt in Index Interrupt and can be applied to a system that has more than one channel assigned as a FIQ. It is built in Index Interrupt compatible with TMS470R1x legacy code.

**Example 15-4. How to Respond to FIQ With Short Latency**

```

        .sect ".intvecs"          ; Interrupt and exception vector sector
00000000h b _RESET                ; RESET interrupt
00000004h b _UNDEF_INST_INT       ; UNDEFINED INSTRUCTION interrupt
00000008h b _SW_INT               ; SOFTWARE interrupt
0000000Ch b _ABORT_PREF_INT      ; ABORT (PREFETCH) interrupt
00000010h b _ABORT_DATA_INT      ; ABORT (DATA) interrupt
00000014h b #-8                  ; Reserved
00000018h b _IRQ_ENTRY_0         ; IRQ interrupt
;*****
; INTERRUPT PROCESSING AREA
;*****
0000001Ch ldrb R8, [PC,#-0x21d]    ; FIQ INTERRUPT ENTRY
; R8 used to get the FIQ index
; with address pointer to the
; first FIQ banked register
00000020h ldr PC, [PC, R8, LSL#2]  ; Branch to the indexed interrupt
; routine. The prefetch
; operation causes the PC to be 2
; words (8 bytes) ahead of the
; current instruction, so
; pointing to _INT_TABLE.
; Required due to pipeline.
00000024h nop
;*****
00000028h _INT_TABLE              ; FIQ INTERRUPT DISPATCH
;*****
0000002Ch .word _FIQ_TABLE        ; beginning of FIQ Dispatch
00000030h .word _ISR1            ; dispatch to interrupt routine 1
00000034h .word _ISR2            ; dispatch to interrupt routine 2
.
.
.

```

Another way to improve the FIQ latency is to assign only one channel to the FIQ interrupt and to map the ISR code corresponding to this channel directly starting at 0x1C.

**NOTE:** When the CPU is in vector-enabled mode, [Example 15-3](#) and [Example 15-4](#) are still valid. The difference is that the CPU will not read from the 0x18 location during IRQ interrupt, but will jump directly to the corresponding ISR routine.

## 15.8 Registers

This section details the VIM module registers, summarized in [Table 15-1](#). 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 for read and write. Write is only possible in privilege mode. The start address of the VIM module is 0xFFFFFE00. The parity related VIM registers are placed at 0xFFFFFDEC~0xFFFFDF8.

**Table 15-1. VIM Control Register Summary**

Offset	Acronym	Register Description	Section
FFFF FDECh	PARFLG	Interrupt Vector Table Parity Flag Register	<a href="#">Section 15.8.1</a>
FFFF FDF0h	PARCTL	Interrupt Vector Table Parity Control Register	<a href="#">Section 15.8.2</a>
FFFF FDF4h	ADDERR	Address Parity Error Register	<a href="#">Section 15.8.3</a>
FFFF FDF8h	FBPARERR	Fall-Back Address Parity Error Register	<a href="#">Section 15.8.4</a>
FFFF FE00h	IRQINDEX	IRQ Index Offset Vector Register	<a href="#">Section 15.8.6</a>
FFFF FE04h	FIQINDEX	FIQ Index Offset Vector Register	<a href="#">Section 15.8.7</a>
FFFF FE10h	FIRQPRO	FIQ/IRQ Program Control Register 0	<a href="#">Section 15.8.8</a>
FFFF FE14h	FIRQPR1	FIQ/IRQ Program Control Register 1	<a href="#">Section 15.8.8</a>
FFFF FE18h	FIRQPR2	FIQ/IRQ Program Control Register 2	<a href="#">Section 15.8.8</a>
FFFF FE20h	INTREQ0	Pending Interrupt Read Location Register 0	<a href="#">Section 15.8.9</a>
FFFF FE24h	INTREQ1	Pending Interrupt Read Location Register 1	<a href="#">Section 15.8.9</a>
FFFF FE28h	INTREQ2	Pending Interrupt Read Location Register 2	<a href="#">Section 15.8.9</a>
FFFF FE30h	REQENASET0	Interrupt Enable Set Register 0	<a href="#">Section 15.8.10</a>
FFFF FE34h	REQENASET1	Interrupt Enable Set Register 1	<a href="#">Section 15.8.10</a>
FFFF FE38h	REQENASET2	Interrupt Enable Set Register 2	<a href="#">Section 15.8.10</a>
FFFF FE40h	REQENACLRO	Interrupt Enable Clear Register 0	<a href="#">Section 15.8.11</a>
FFFF FE44h	REQENACLRO1	Interrupt Enable Clear Register 1	<a href="#">Section 15.8.11</a>
FFFF FE48h	REQENACLRO2	Interrupt Enable Clear Register 2	<a href="#">Section 15.8.11</a>
FFFF FE50h	WAKEENASET0	Wake-up Enable Set Register 0	<a href="#">Section 15.8.12</a>
FFFF FE54h	WAKEENASET1	Wake-up Enable Set Register 1	<a href="#">Section 15.8.12</a>
FFFF FE58h	WAKEENASET2	Wake-up Enable Set Register 2	<a href="#">Section 15.8.12</a>
FFFF FE60h	WAKEENACLRO	Wake-up Enable Clear Register 0	<a href="#">Section 15.8.13</a>
FFFF FE64h	WAKEENACLRO1	Wake-up Enable Clear Register 1	<a href="#">Section 15.8.13</a>
FFFF FE68h	WAKEENACLRO2	Wake-up Enable Clear Register 2	<a href="#">Section 15.8.13</a>
FFFF FE70h	IRQVECREG	IRQ Interrupt Vector Register	<a href="#">Section 15.8.14</a>
FFFF FE74h	FIQVECREG	FIQ Interrupt Vector Register	<a href="#">Section 15.8.15</a>
FFFF FE78h	CAPEVT	Capture Event Register	<a href="#">Section 15.8.16</a>
FFFF FE80h - FFFF FEDCh	CHANCTRL	VIM Interrupt Control Register	<a href="#">Section 15.8.17</a>

### 15.8.1 Interrupt Vector Table Parity Flag Register (PARFLG)

Figure 15-11 and Table 15-2 describe this register.

**Figure 15-11. Interrupt Vector Table Parity Flag Register (PARFLG) [Offset = FFFF FDECh]**

31	Reserved	16
R-0		
15	Reserved	1 0
R-0		PARFLG R/WP-0

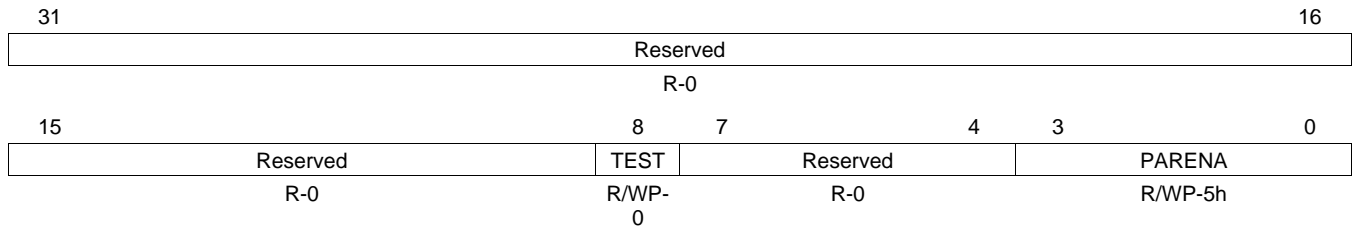
LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; -n = value after reset

**Table 15-2. Interrupt Vector Table Parity Flag Register (PARFLG) Field Descriptions**

Bit	Field	Value	Description
31-15	Reserved	0	Read returns 0. Writes have no effect.
0	PARFLG	0	<i>Read:</i> No parity error has occurred. <i>Write:</i> A write to this bit has no effect.
		1	<i>Read:</i> A parity error has occurred and the Interrupt Vector Table is bypassed. <i>Write:</i> The PARFLG is cleared and the interrupt vector can be read from the Interrupt Vector Table.

## 15.8.2 Interrupt Vector Table Parity Control Register (PARCTL)

**Figure 15-12. Interrupt Vector Table Parity Control Register (PARCTL) [Offset = FFFF FDF0h]**



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

**Table 15-3. Interrupt Vector Table Parity Control Register (PARCTL) Field Descriptions**

Bit	Field	Value	Description
31-9	Reserved	0	Read returns 0. Writes have no effect.
8	TEST	0	This bit maps the parity bits into the Interrupt Vector Table frame to make them accessible by the CPU. Parity bits are not memory mapped.
		1	Parity bits are memory mapped.
7-4	Reserved	0	Read returns 0. Writes have no effect.
3-0	PARENA	5h	VIM parity enable. The VIM parity is disabled.
		All Others	The VIM parity is enabled. <b>Note: To avoid soft error to disable VIM parity checking, it is recommended to write Ah to enable parity checking.</b>



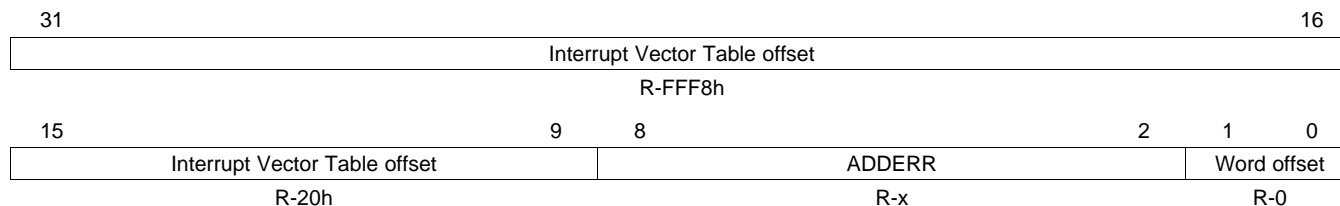
### 15.8.3 Address Parity Error Register (ADDERR)

The address parity error register gives the address of the first parity error location.

**NOTE:** No computation is needed when reading the complete register to retrieve the address in the Interrupt Vector Table.

**NOTE:** This register will never be reset by a power-on reset nor any other reset source.

**Figure 15-13. Address Parity Error Register (ADDERR) [Offset = FFFF FDF4h]**



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

**Table 15-4. Address Parity Error Register (PARERR) Field Descriptions**

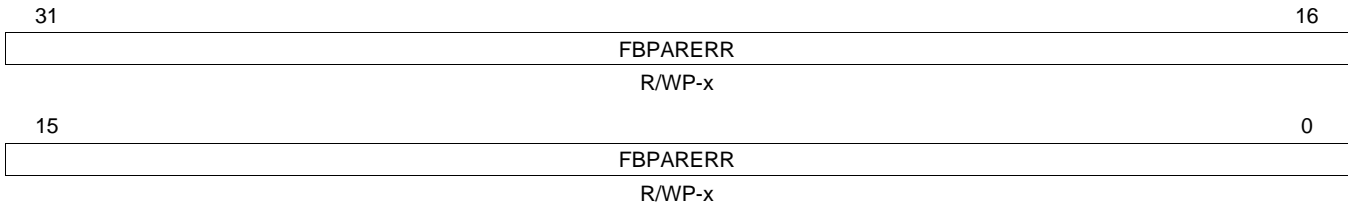
Bit	Field	Value	Description
31-9	Interrupt Vector Table offset		Interrupt Vector Table offset. Reads are always FF F820h; writes have no effect
8-2	ADDERR		Address parity error register. This register gives the address of the first encountered parity error since the flag has been clear. Subsequent parity errors will not update this register until the PARFLG register has been cleared.  Note: This register is valid only when PARFLG is set (see <a href="#">Section 15.8.1</a> ).
1-0	Word offset		Word offset. Reads are always 0; writes have no effect.

### 15.8.4 Fall-Back Address Parity Error Register (FBPARERR)

This register provides a fall-back address to the VIM if a parity error has occurred in the Interrupt Vector Table. [Figure 15-14](#) and [Table 15-5](#) describe this register.

**NOTE:** This register will never be reset by a power-on reset nor any other reset source.

**Figure 15-14. Fall-Back Address Parity Error Register (FBPARERR) [Offset = FFFF FDF8h]**



LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; -n = value after reset; x = Indeterminate

**Table 15-5. Fall Back Address Parity Error Register (FBPARERR) Field Descriptions**

Bit	Field	Value	Description
31-0	FBPARERR	0-FFFF FFFFh	Fall back address parity error. This register is used by the VIM if the Interrupt Vector Table has been corrupted. The contents of the IRQVECREG and FIQVECREG registers will reflect the value programmed in FBPARERR. The value provided to the VIC port will also reflect FBPARERR until the PARFLG register has been cleared.  This register provides the address of the ISR that will restore the integrity of the Interrupt Vector Table.

### 15.8.5 VIM Offset Vector Registers

The VIM offset register provides the user with the numerical index value that represents the pending interrupt with the highest precedence. The register IRQINDEX holds the index to the highest priority IRQ interrupt; the register FIQINDEX holds the index to the highest priority FIQ interrupt. The index can be used to locate the interrupt routine in a dispatch table, as shown in [Table 15-6](#).

**Table 15-6. Interrupt Dispatch**

IRQINDEX / FIQINDEX Register Bit Field	Highest Priority Pending Interrupt Enabled
0x00	No interrupt
0x01	Channel 0
:	:
0x59	Channel 94
0x60	Channel 95

---

**NOTE:** Channel 95 has no dedicated interrupt vector table entry. Therefore, Channel 95 shall NOT be used in application.

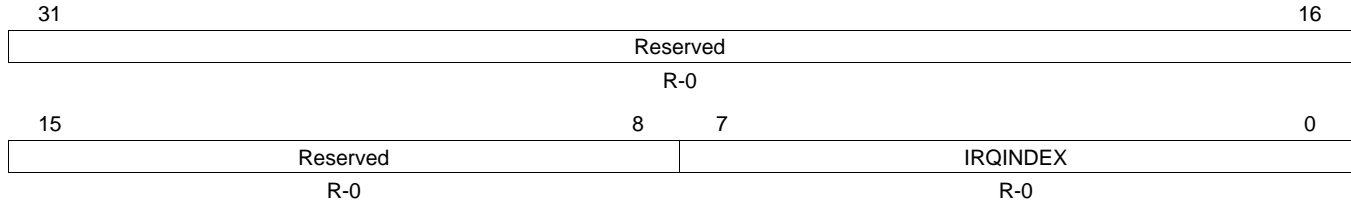
---

The VIM offset registers are read only. They are updated continuously by the VIM. When an interrupt is serviced, the offset vectors show the index for the next highest pending interrupt or 0x0 if no interrupt is pending.

### 15.8.6 IRQ Index Offset Vector Register (IRQINDEX)

The IRQ offset register provides the user with the numerical index value that represents the pending IRQ interrupt with the highest priority. [Figure 15-15](#) and [Table 15-7](#) describe this register.

**Figure 15-15. IRQ Index Offset Vector Register (IRQINDEX) [Offset = FFFF FE00h]**



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

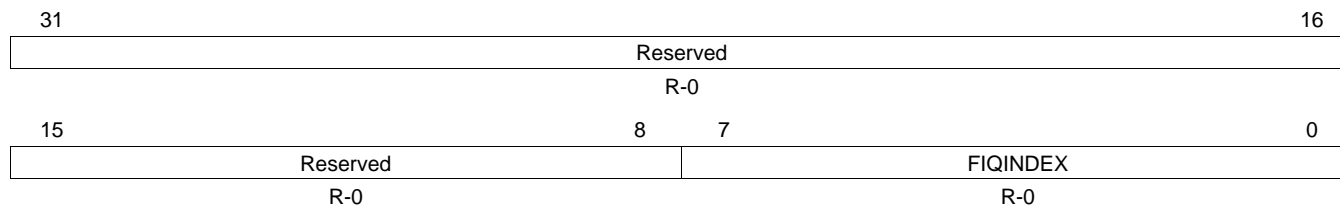
**Table 15-7. IRQ Index Offset Vector Register (IRQINDEX) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Read returns 0. Writes have no effect.
7-0	IRQINDEX	0-FFh	<b>IRQ index vector.</b> The least significant bits represent the index of the IRQ pending interrupt with the highest precedence, as shown in <a href="#">Table 15-6</a> . When no interrupts are pending, the least significant byte of IRQINDEX is 0.

### 15.8.7 FIQ Index Offset Vector Registers (FIQINDEX)

The FIQINDEX register provides the user with a numerical index value that represents the pending FIQ interrupt with the highest priority. [Figure 15-16](#) and [Table 15-8](#) describe this register.

**Figure 15-16. FIQ Index Offset Vector Register (FIQINDEX) [Offset = FFFF FE04h]**



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

**Table 15-8. FIQ Index Offset Vector Register (FIQINDEX) Field Descriptions**

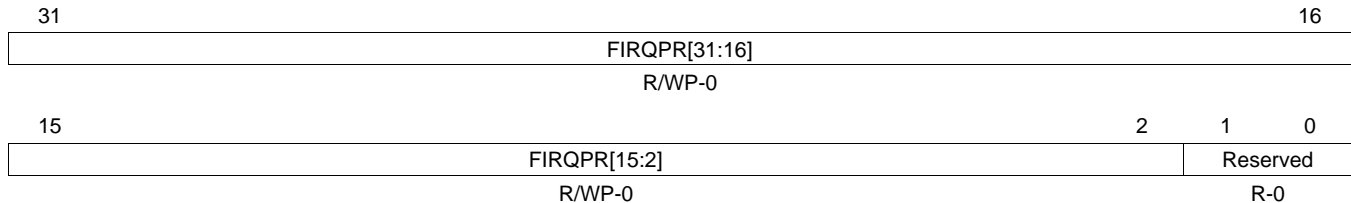
Bit	Field	Value	Description
31-8	Reserved	0	Read returns 0. Writes have no effect.
7-0	FIQINDEX	0-FFh	<b>FIQ index offset vector.</b> The least significant bits represent the index of the FIQ pending interrupt with the highest precedence, as shown in <a href="#">Table 15-6</a> . When no interrupts are pending, the least significant byte of FIQINDEX is 0x00.

### 15.8.8 FIQ/IRQ Program Control Registers[0] (FIRQPR[0])

The FIQ/IRQ program control registers (FIRQPR[0]) determine whether a given interrupt request will be either FIQ or IRQ. [Figure 15-17](#), [Figure 15-18](#), [Figure 15-19](#) and [Table 15-9](#) describe these registers.

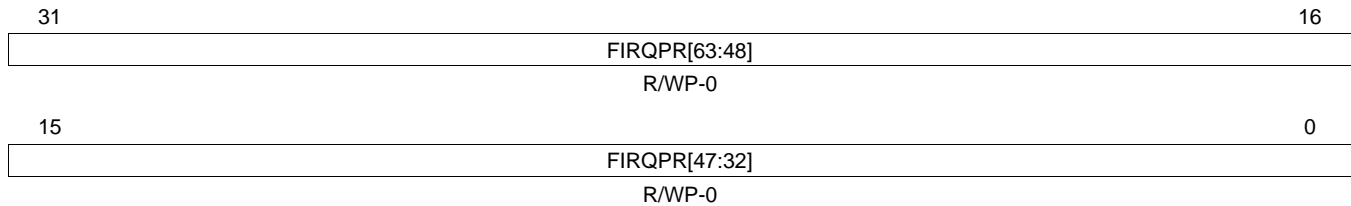
**NOTE:** Channel 0 and 1 are FIQ only, not impacted by this register.

**Figure 15-17. FIQ/IRQ Program Control Register 0 (FIRQPR0) [offset = FFFF FE10h]**



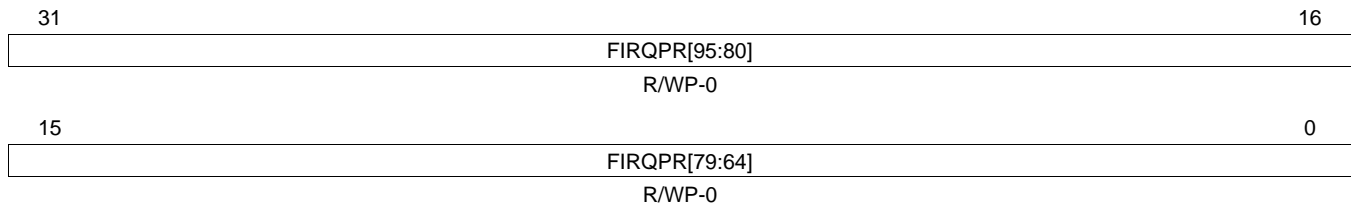
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Figure 15-18. FIQ/IRQ Program Control Register 1 (FIRQPR1) [offset = FFFF FE14h]**



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

**Figure 15-19. FIQ/IRQ Program Control Register 2 (FIRQPR2) [offset = FFFF FE18h]**



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

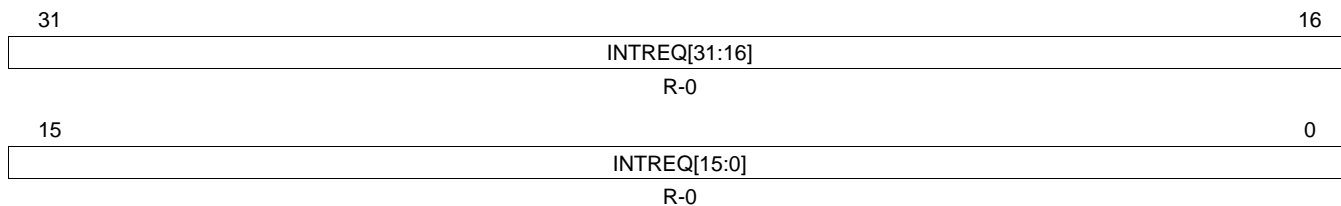
**Table 15-9. FIQ/IRQ Program Control Registers[0] (FIRQPR[0]) Field Descriptions**

Bit	Field	Value	Description
95-2	FIRQPR[95:2]	0	<b>FIQ/IRQ program control bits.</b> These bits determine whether an interrupt request from a peripheral is of type FIQ or IRQ. Bit FIRQPRx[95:2] corresponds to request channel[95:2]. Interrupt request is of IRQ type.
		1	
1-0	Reserved	0	Read only. Writes have no effect.

### 15.8.9 Pending Interrupt Read Location Registers[0] (INTREQ[0])

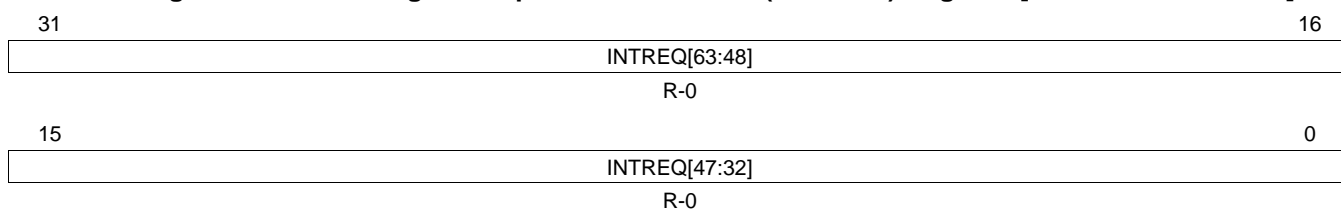
The pending interrupt register gives the pending interrupt requests. The register is updated every vbus clock cycle. Figure 15-20, Figure 15-21, Figure 15-22 and Table 15-10 describe this register.

**Figure 15-20. Pending Interrupt Read Location Register 0 (INTREQ0) [offset = FFFF FE20h]**



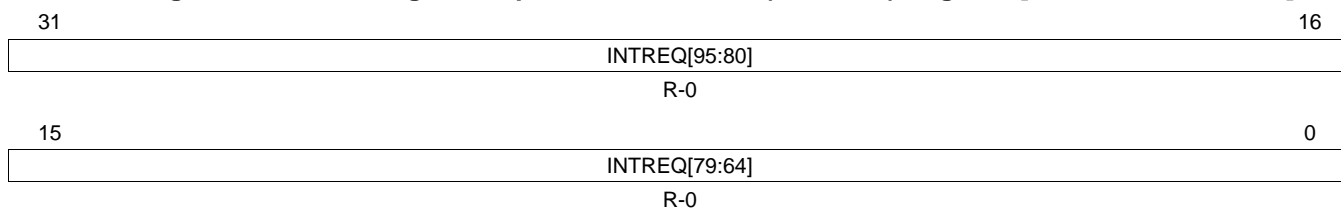
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Figure 15-21. Pending Interrupt Read Location 1 (INTREQ1) Register [offset = FFFF FE24h]**



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

**Figure 15-22. Pending Interrupt Read Location 2 (INTREQ2) Register [offset = FFFF FE28h]**



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

**Table 15-10. Pending Interrupt Read Registers[0] (INTREQ[0]) Field Descriptions**

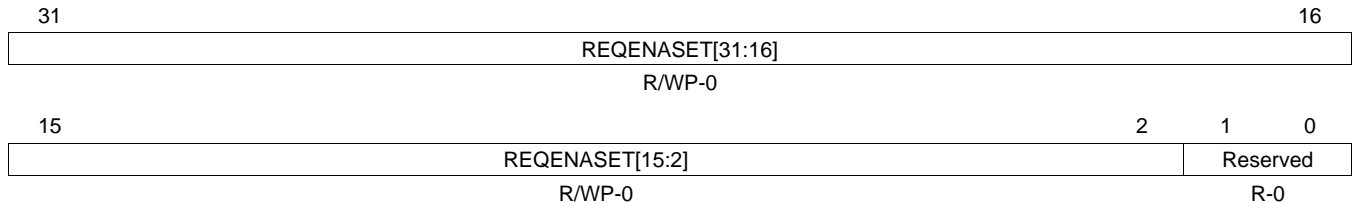
Bit	Field	Value	Description
95-0	FIRQPR[95:0]		Pending interrupt bits. These bits determine whether an interrupt request is pending for the request channel between 0 and 95. The interrupt ENABLE register does not affect the value of the interrupt pending bit. Bit INTREQx[95:0] corresponds to request channel[95:0].
		0	No interrupt event has occurred.
		1	An interrupt is pending.

### 15.8.10 Interrupt Enable Set Registers[0] (REQENASET[0])

The interrupt register enable selectively enables individual request channels. [Figure 15-23](#), [Figure 15-24](#), [Figure 15-25](#) and [Table 15-11](#) describe these registers.

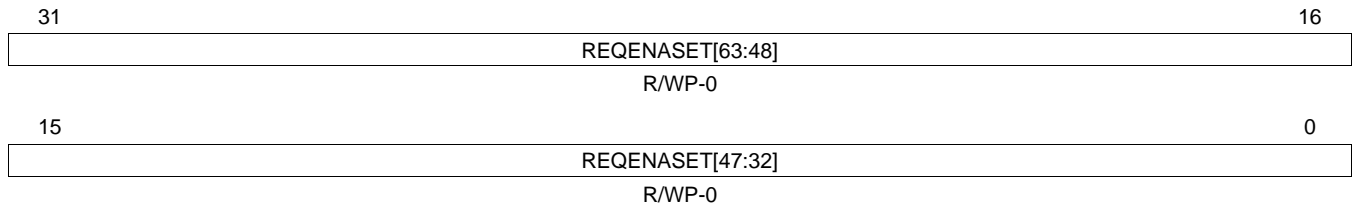
**NOTE:** Channel 0 and 1 are always enabled, not impacted by this register.

**Figure 15-23. Interrupt Enable Set Register 0 (REQENASET0) [offset = FFFF FE30h]**



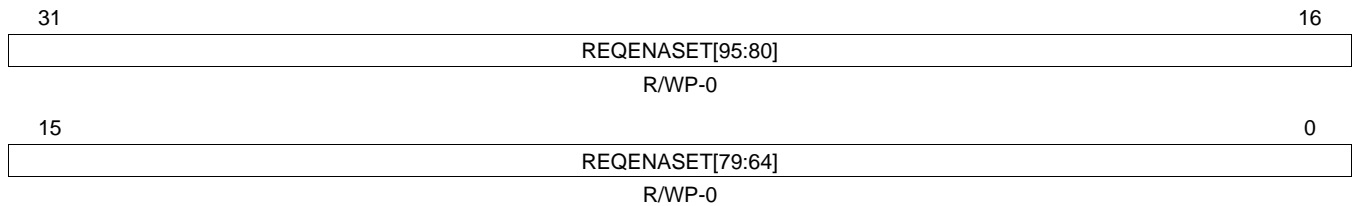
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Figure 15-24. Interrupt Enable Set Register 1 (REQENASET1) [offset = FFFF FE34h]**



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

**Figure 15-25. Interrupt Enable Set Register 2 (REQENASET2) [offset = FFFF FE38h]**



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

**Table 15-11. Interrupt Enable Set Register[0] (REQENASET[0]) Field Descriptions**

Bit	Field	Value	Description
95-2	REQENASET[95:2]	0	<b>Request enable set bits.</b> This vector determines whether the interrupt request channel is enabled. Bit REQENASETx[95:2] corresponds to request channel[95:2]. <i>Read:</i> Interrupt request channel is disabled. <i>Write:</i> A write of 0 has no effect.
		1	
1-0	Reserved	0	Read only. Writes have no effect.

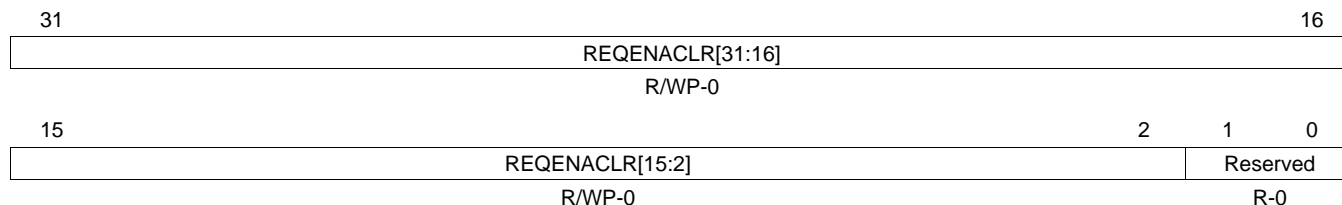


### 15.8.11 Interrupt Enable Clear Registers[0] (REQENACLR[0])

The interrupt register enable selectively disables individual request channels. Figure 15-26, Figure 15-27, Figure 15-28 and Table 15-12 describe these registers.

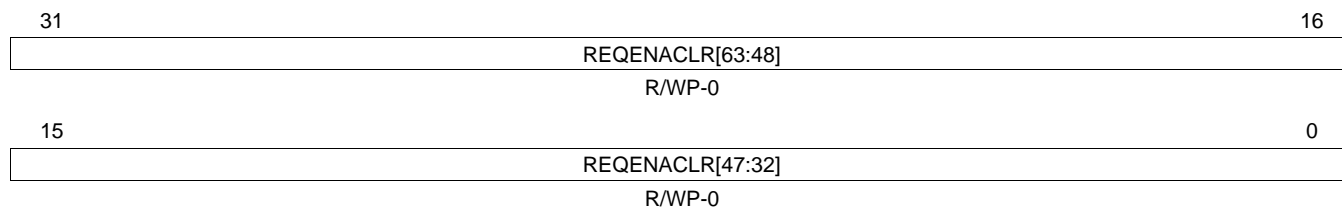
**NOTE:** Channel 0 and 1 are always enabled, not impacted by this register.

**Figure 15-26. Interrupt Enable Clear Register 0 (REQENACLR0) [offset = FFFF FE40h]**



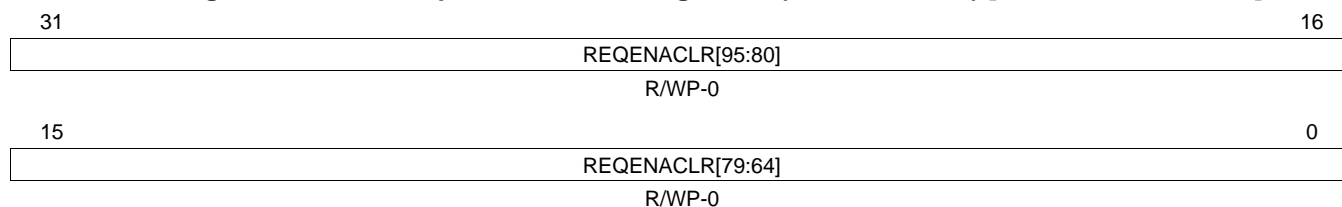
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Figure 15-27. Interrupt Enable Clear Register 1 (REQENACLR1) [offset = FFFF FE44h]**



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

**Figure 15-28. Interrupt Enable Clear Register 2 (REQENACLR2) [offset = FFFF FE48h]**



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

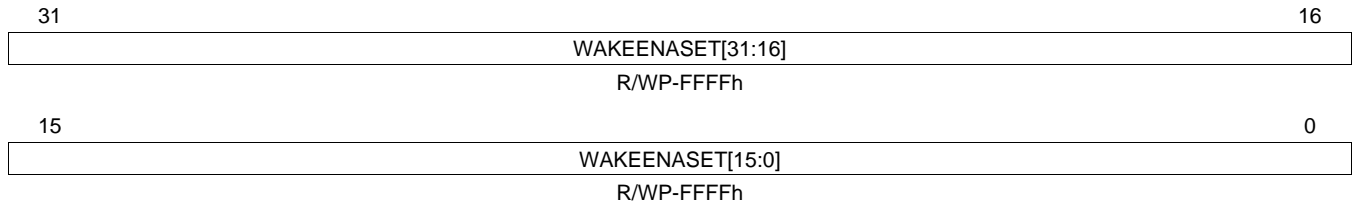
**Table 15-12. Request Enable Clear Register (REQENACLR) Field Descriptions**

Bit	Field	Value	Description
95-2	REQENACLR[95:2]	0	<b>Request enable clear bits.</b> This vector determines whether the interrupt request channel is enabled. Bit REQENACLRx[95:2] corresponds to request channel[95:2]. <i>Read:</i> Interrupt request channel is disabled. <i>Write:</i> A write of 0 has no effect.
		1	<i>Read:</i> The interrupt request channel is enabled. <i>Write:</i> The interrupt request channel is disabled.
1-0	Reserved	0	Read only. Writes have no effect.

**15.8.12 Wake-Up Enable Set Registers[0] (WAKEENASET[0])**

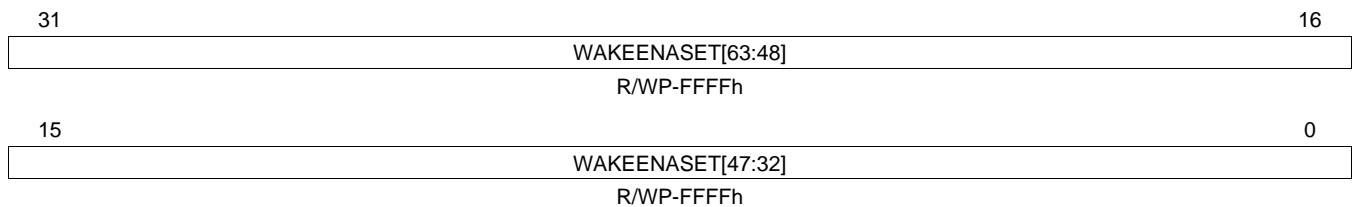
The wake-up enable registers selectively enables individual wake-up interrupt request lines. [Figure 15-29](#), [Figure 15-30](#), [Figure 15-31](#) and [Table 15-13](#) describe these registers.

**Figure 15-29. Wake-Up Enable Set Register 0 (WAKEENASET0) [offset = FFFF FE50h]**



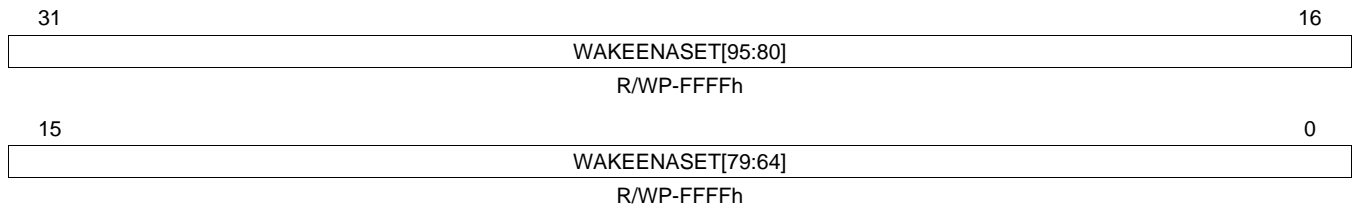
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Figure 15-30. Wake-Up Enable Set Register 1 (WAKEENASET1) [offset = FFFF FE54h]**



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

**Figure 15-31. Wake-Up Enable Set Register 2 (WAKEENASET2) [offset = FFFF FE58h]**



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

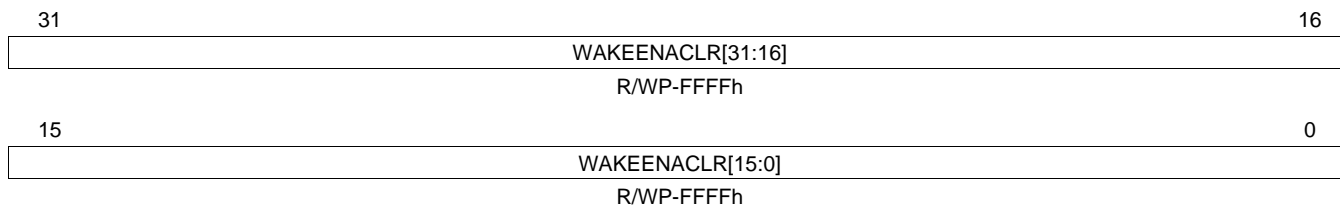
**Table 15-13. Wake-Up Enable Set Registers[0] (WAKEENASET[0]) Field Descriptions**

Bit	Field	Value	Description
95-0	WAKEENASET[95:0]	0	Wake-up enable set bits. This vector determines whether the wake-up interrupt line is enabled. Bit WAKEENASETx[95:0] corresponds to interrupt request channel[95:0]. <i>Read:</i> Interrupt request channel is disabled. <i>Write:</i> A write of 0 has no effect.
		1	<i>Read or Write:</i> The interrupt request channel is enabled.

### 15.8.13 Wake-Up Enable Clear Registers[0] (WAKEENACLR[0])

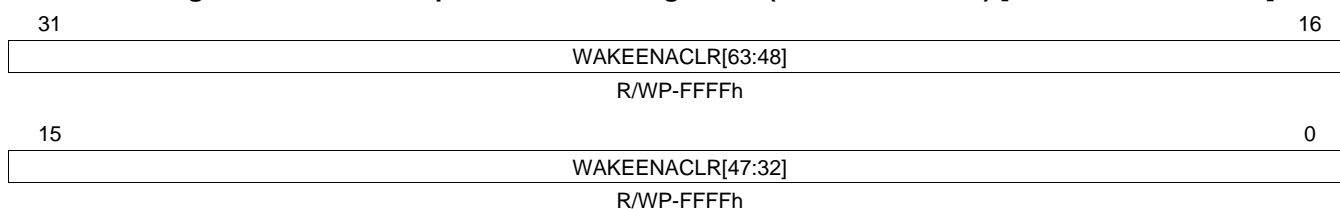
The wake-up enable register selectively disables individual wake-up interrupt request lines. [Figure 15-32](#), [Figure 15-33](#), [Figure 15-34](#) and [Table 15-14](#) describe these registers.

**Figure 15-32. Wake-Up Enable Clear Register 0 (WAKEENACLR0) [offset = FFFF FE60h]**



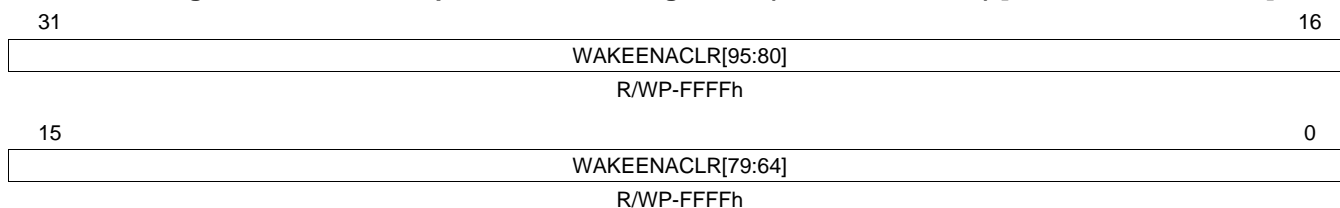
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Figure 15-33. Wake-Up Enable Clear Register 1 (WAKEENACLR1) [offset = FFFF FE64h]**



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

**Figure 15-34. Wake-Up Enable Clear Register 2 (WAKEENACLR2) [offset = FFFF FE68h]**



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

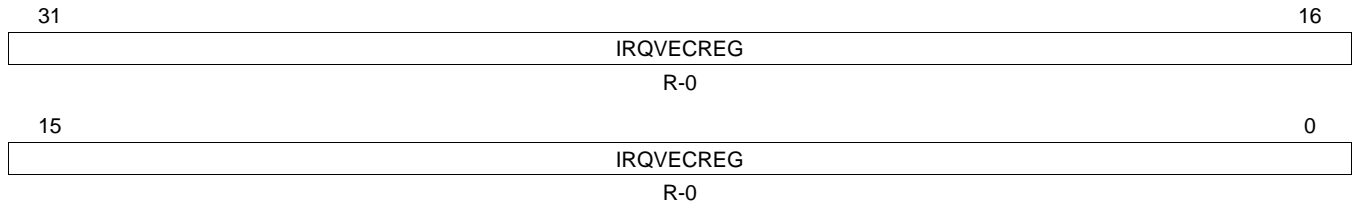
**Table 15-14. Wake-Up Enable Clear Registers[0] (WAKEENACLR[0]) Field Descriptions**

Bit	Field	Value	Description
95-0	WAKEENACLR[95:0]	0	Wake-up enable clear bits. This vector determines whether the wake-up interrupt line is enabled. Bit WAKEENACLRx[95:0] corresponds to interrupt request channel[95:0]. <i>Read:</i> Wake-up interrupt channel is disabled. <i>Write:</i> A write of 0 has no effect.
		1	<i>Read:</i> The wake-up interrupt channel is enabled. <i>Write:</i> The wake-up interrupt channel is disabled.

**15.8.14 IRQ Interrupt Vector Register (IRQVECREG)**

The interrupt vector register gives the address of the enabled and active IRQ interrupt. [Figure 15-35](#) and [Table 15-15](#) describe these registers.

**Figure 15-35. IRQ Interrupt Vector Register (IRQVECREG) [Offset = FFFF FE70h]**



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

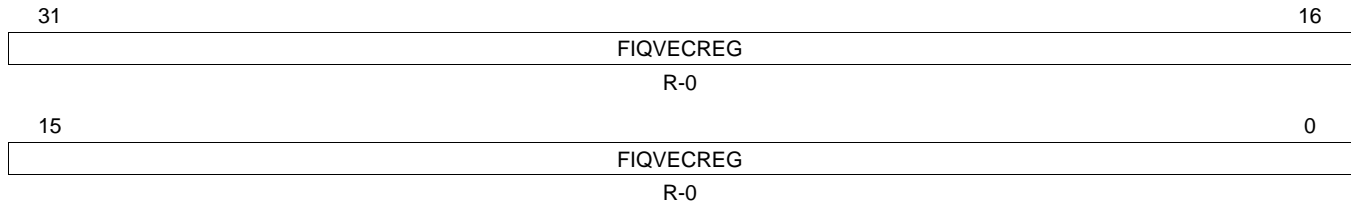
**Table 15-15. IRQ Interrupt Vector Register (IRQVECREG) Field Descriptions**

Bit	Field	Value	Description
31-0	IRQVECREG	From <a href="#">Section 15.4</a>	IRQ interrupt vector register. This vector gives the address of the ISR with the highest pending IRQ request. The CPU reads the address and branches to this location.

### 15.8.15 FIQ Interrupt Vector Register (FIQVECREG)

The interrupt vector register gives the address of the enabled and active FIQ interrupt. [Figure 15-36](#) and [Table 15-16](#) describe these registers.

**Figure 15-36. IRQ Interrupt Vector Register (FIQVECREG) [Offset = FFFF FE74h]**



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

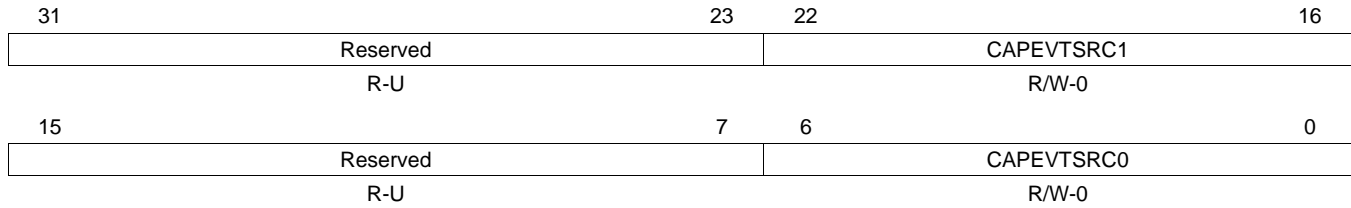
**Table 15-16. FIQ Interrupt Vector Register (FIQVECREG) Field Descriptions**

Bit	Field	Value	Description
31-0	FIQVECREG	From <a href="#">Section 15.4</a>	FIQ interrupt vector register. This vector gives the address of the ISR with the highest pending FIQ request. The CPU reads the address and branches to this location.

### 15.8.16 Capture Event Register (CAPEVT)

Figure 15-37 and Table 15-17 describe this register.

**Figure 15-37. Capture Event Register (CAPEVT) [Offset = FFFF FE78h]**



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

**Table 15-17. Capture Event Register (CAPEVT) Field Descriptions**

Bit	Field	Value	Description	
31-23	Reserved	0	Reads are indeterminate and writes have no effect.	
22-16	CAPEVTSRC1[	0	Capture event source 1 mapping control. These bits determine which interrupt request maps to the capture event source 1 of the RTI:	
		1h		Interrupt request 1.
		...		...
		5Eh		Interrupt request 63.
15-7	Reserved	0	Reads are indeterminate and writes have no effect.	
6-0	CAPEVTSRC0	0	Capture event source 0 mapping control. These bits determine which interrupt request maps to the capture event source 0 of the RTI:	
		1h		Interrupt request 1.
		...		...
		5Fh		Interrupt request 63.

### 15.8.17 VIM Interrupt Control Register[0:31] (CHANCTRL[0:23])

Twenty four interrupt control registers control the 96 interrupt channels of the VIM. Each register controls four interrupt channels: each of them is indexed from 0 to 95. Table 15-18 shows the organization of all the registers and the reset value of each. Each four fields of the register has been named with a generic index that refers to the detailed register organization. Figure 15-38 and Table 15-19 describe these registers.

**Table 15-18. Interrupt Control Registers Organization**

Offset Address <sup>(1)</sup>	Register Mnemonic	Register Field 31:24 CHANMAP <sub>x<sub>0</sub></sub>	Register Field 23:16 CHANMAP <sub>x<sub>1</sub></sub>	Register Field 15:8 CHANMAP <sub>x<sub>2</sub></sub>	Register Field 7:0 CHANMAP <sub>x<sub>3</sub></sub>	Resetvalue
0xFFFFFE80	CHANCTRL0	CHANMAP0	CHANMAP1	CHANMAP2	CHANMAP3	0x00010203
0xFFFFFE84	CHANCTRL1	CHANMAP4	CHANMAP5	CHANMAP6	CHANMAP7	0x04050607
...	...	...	...	...	...	...
0xFFFFFED8	CHANCTRL22	CHANMAP88	CHANMAP89	CHANMAP90	CHANMAP91	0x58595A5B
0xFFFFFEDC	CHANCTRL23	CHANMAP92	CHANMAP93	CHANMAP94	CHANMAP95	0x5C5D5E5F

**NOTE:** CHANMAP0 and CHANMAP1 are not programmable. CHAN0 and CHAN1 are hard wired to INT\_REQ0 and INT\_REQ1.

**NOTE:** Do NOT write any value other than 0x5F to CHANMAP95. Channel 95 is reserved because no interrupt vector table entry supports this channel.

**Figure 15-38. Interrupt Control Registers[0:31] (CHANCTRL[0:23]) [Offset = FFFF FE80h - FFFF FEDCh]**

31	30	24	23	22	16
Rsvd.	CHANMAP <sub>x<sub>0</sub></sub>			Rsvd.	CHANMAP <sub>x<sub>1</sub></sub>
R-U	R/W-n			R-U	R/W-n
15	14	8	7	6	0
Rsvd.	CHANMAP <sub>x<sub>2</sub></sub>			Rsvd.	CHANMAP <sub>x<sub>3</sub></sub>
R-U	R/W-n			R-U	R/W-n

LEGEND: R/W = Read/Write; R = Read only; U = Undefined; -n = value after reset (see Table 15-18)

**Table 15-19. Interrupt Control Registers[0:31] (CHANCTRL[0:15]) Field Descriptions**

Bit	Field	Value	Description
31	Reserved	0	Reads are indeterminate and writes have no effect.
30-24	CHANMAP <sub>x<sub>0</sub></sub>	0	CHANMAP <sub>x<sub>0</sub></sub> (6-0). Interrupt CHAN <sub>x<sub>0</sub></sub> mapping control. These bits determine which interrupt request the priority channel CHAN <sub>x<sub>0</sub></sub> maps to: <i>Read:</i> Interrupt request 0 maps to channel priority CHAN <sub>x<sub>0</sub></sub> . <i>Write:</i> The default value of this bit after reset is given in Table 15-18. The channel priority CHAN <sub>x<sub>0</sub></sub> is set with the interrupt request.
		1h	<i>Read:</i> Interrupt request 1 maps to channel priority CHAN <sub>x<sub>0</sub></sub> . <i>Write:</i> The default value of this bit after reset is given in Table 15-18. The channel priority CHAN <sub>x<sub>0</sub></sub> is set with the interrupt request.
		...	...
		5Fh	<i>Read:</i> Interrupt request 63 maps to channel priority CHAN <sub>x<sub>0</sub></sub> . <i>Write:</i> The default value of this bit after reset is given in Table 15-18. The channel priority CHAN <sub>x<sub>0</sub></sub> is set with the interrupt request.
23	Reserved	0	Reads are indeterminate and writes have no effect.

**Table 15-19. Interrupt Control Registers[0:31] (CHANCTL[0:15] Field Descriptions (continued))**

Bit	Field	Value	Description
22-16	CHANMAPx <sub>1</sub>	<p>0</p> <p>1h</p> <p>...</p> <p>5Fh</p>	<p>CHANMAPx<sub>1</sub>(6-0). Interrupt CHANx<sub>1</sub> mapping control. These bits determine which interrupt request the priority channel CHANx<sub>1</sub> maps to:</p> <p><i>Read:</i> Interrupt request 0 maps to channel priority CHANx<sub>1</sub>.</p> <p><i>Write:</i> The default value of this bit after reset is given in <a href="#">Table 15-18</a>. The channel priority CHANx<sub>1</sub> is set with the interrupt request.</p> <p><i>Read:</i> Interrupt request 1 maps to channel priority CHANx<sub>1</sub>.</p> <p><i>Write:</i> The default value of this bit after reset is given in <a href="#">Table 15-18</a>. The channel priority CHANx<sub>1</sub> is set with the interrupt request.</p> <p>...</p> <p><i>Read:</i> Interrupt request 63 maps to channel priority CHANx<sub>1</sub>.</p> <p><i>Write:</i> The default value of this bit after reset is given in <a href="#">Table 15-18</a>. The channel priority CHANx<sub>1</sub> is set with the interrupt request.</p>
15	Reserved	0	Reads are indeterminate and writes have no effect.
14-8	CHANMAPx <sub>2</sub>	<p>0</p> <p>1h</p> <p>...</p> <p>5Fh</p>	<p>CHANMAPx<sub>2</sub>(6-0). Interrupt CHANx<sub>2</sub> mapping control. These bits determine which interrupt request the priority channel CHANx<sub>2</sub> maps to:</p> <p><i>Read:</i> Interrupt request 0 maps to channel priority CHANx<sub>2</sub>.</p> <p><i>Write:</i> The default value of this bit after reset is given in <a href="#">Table 15-18</a>. The channel priority CHANx<sub>2</sub> is set with the interrupt request.</p> <p><i>Read:</i> Interrupt request 1 maps to channel priority CHANx<sub>2</sub>.</p> <p><i>Write:</i> The default value of this bit after reset is given in <a href="#">Table 15-18</a>. The channel priority CHANx<sub>2</sub> is set with the interrupt request.</p> <p>...</p> <p><i>Read:</i> Interrupt request 63 maps to channel priority CHANx<sub>2</sub>.</p> <p><i>Write:</i> The default value of this bit after reset is given in <a href="#">Table 15-18</a>. The channel priority CHANx<sub>2</sub> is set with the interrupt request.</p>
7	Reserved	0	Reads are indeterminate and writes have no effect.
6-0	CHANMAPx <sub>3</sub>	<p>0</p> <p>1h</p> <p>...</p> <p>5Fh</p>	<p>CHANMAPx<sub>3</sub>(6-0). Interrupt CHANx<sub>3</sub> mapping control. These bits determine which interrupt request the priority channel CHANx<sub>3</sub> maps to:</p> <p><i>Read:</i> Interrupt request 0 maps to channel priority CHANx<sub>3</sub>.</p> <p><i>Write:</i> The default value of this bit after reset is given in <a href="#">Table 15-18</a>. The channel priority CHANx<sub>3</sub> is set with the interrupt request.</p> <p><i>Read:</i> Interrupt request 1 maps to channel priority CHANx<sub>3</sub>.</p> <p><i>Write:</i> The default value of this bit after reset is given in <a href="#">Table 15-18</a>. The channel priority CHANx<sub>3</sub> is set with the interrupt request.</p> <p>...</p> <p><i>Read:</i> Interrupt request 63 maps to channel priority CHANx<sub>3</sub>.</p> <p><i>Write:</i> The default value of this bit after reset is given in <a href="#">Table 15-18</a>. The channel priority CHANx<sub>3</sub> is set with the interrupt request.</p>





---

---

## ***Direct Memory Access Controller (DMA) Module***

---

---

This chapter describes the direct memory access (DMA) controller.

<b>Topic</b>	<b>Page</b>
<b>16.1 Overview .....</b>	<b>640</b>
<b>16.2 Module Operation .....</b>	<b>641</b>
<b>16.3 Control Registers and Control Packets .....</b>	<b>658</b>

## 16.1 Overview

The DMA controller is used to transfer data between two locations in the memory map in the background of CPU operations. Typically, the DMA is used to:

- Transfer blocks of data between external and internal data memories
- Restructure portions of internal data memory
- Continually service a peripheral
- Page program sections to internal program memory

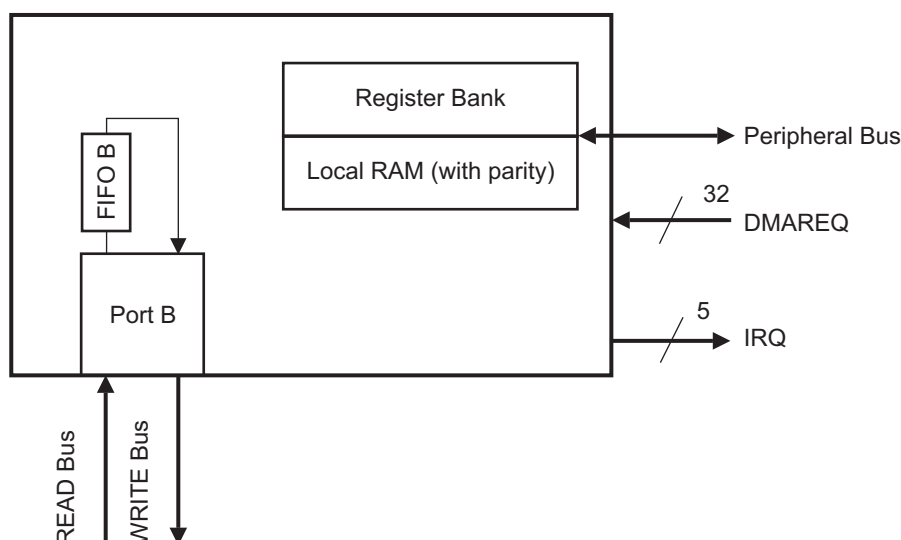
### 16.1.1 Main Features

- CPU independent data transfer
- One master port - PortB (64 bits wide) that interfaces microcontrollers Memory System.
- FIFO buffer (4 entries deep and each 64bit wide)
- Channel control information is stored in RAM protected by parity
- 16 channels with individual enable
- Channel chaining capability
- 32 peripheral DMA requests
- Hardware and Software DMA requests
- 8, 16, 32 or 64-bit transactions supported
- Multiple addressing modes for source/destination (fixed, increment, offset)
- Auto-initiation
- Power-management mode
- Memory Protection for the address range DMA can access with four configurable memory regions

#### 16.1.1.1 Block Diagram

Figure 16-1 gives a detailed view of the DMA internal architecture. DMA data read and write access happens through Port B. FIFO B is 4 levels deep and 64-bits wide. 32 DMA requests go into the DMA that can trigger DMA transfers. Five interrupt request lines go out of the DMA to signal that a certain transfer status is reached. Register banks hold the memory mapped DMA configuration registers. Local RAM consists of DMA control packets and is secured by parity. All the programming / configuration of the DMA controller is done via the Peripheral bus.

**Figure 16-1. DMA Block Diagram**



## 16.2 Module Operation

The DMA acts as an independent master in the platform architecture. All DMA memory and register accesses are performed in user mode. If the DMA writes to registers which are only accessible in privileged mode, the write will not be performed.

The DMA registers and its local RAM can only be accessed in privilege mode. Therefore, it is not possible for the DMA to reprogram itself. Due to the decoupling of the read and write bus, it is possible to execute a maximum of two transfers in parallel, if source and destination of the transfer are not on the same bus system.

### 16.2.1 Memory Space

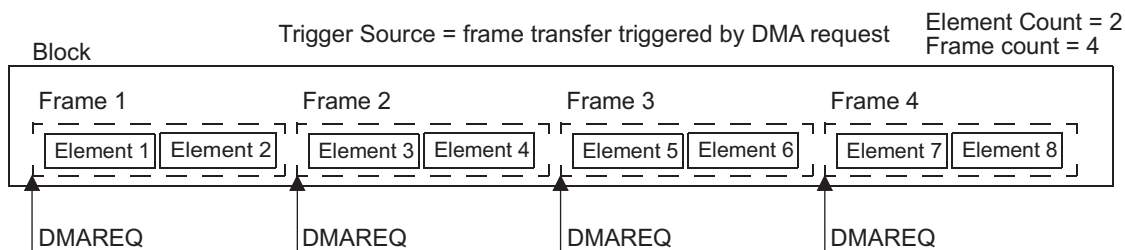
The DMA controller makes no distinction between program memory and data memory. The DMA controller can transfer to and from any space within the 4 Gbyte physical address map, by programming the absolute address for the source and destination in the control packet.

### 16.2.2 DMA Data Access

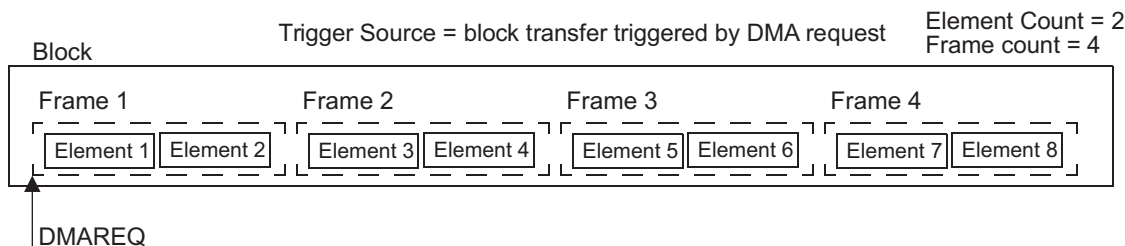
The DMA controller refers to data in three levels of granularity:

- **Element:** Depending on the programmed data type, an 8-bit, 16-bit, 32-bit, or a 64-bit value. The type can be individually selected for the source (read) and destination (write). See [Figure 16-2](#) and [Figure 16-3](#) for an example of the use of elements. An element transfer cannot be interrupted.
- **Frame:** One or more elements to be transferred as a unit. A frame transfer can be interrupted between element transfers. See [Figure 16-2](#) for an example.
- **Block:** One or more frames to be transferred as a unit. Each channel can transfer one block of data (once or multiple times). See [Figure 16-3](#) for an example.

**Figure 16-2. Example of a DMA Transfer Using Frame Trigger Source**



**Figure 16-3. Example of a DMA Transfer Using Block Trigger Source**



### 16.2.3 Addressing Modes

There are three addressing modes supported by the DMA controller that can be setup independent for the source and the destination address:

- Constant -- source and/or destination addresses do not change.
- Post incremented -- source and/or destination address are post-incremented by the element size.
- Indexed -- source and/or destination address is post-incremented as defined in the Element Index Offset Register ([Section 16.3.2.5](#)) and the Frame Index Offset Register ([Section 16.3.2.6](#)).

An unaligned address with respect to the element size is not supported. An address chosen during transfer must be divisible by the element size.

### 16.2.4 DMA Channel Control Packets

There are a total of 16 control packets. Each control packet is associated with a channel in a fixed order. For example, control packet 0 stores channel information for channel 0. The DMA requests can be mapped to the individual channels as described in [Section 16.2.7](#). [Figure 16-4](#) illustrates the mapping scheme between DMA requests and channels. Each control packet contains nine fields. The first six fields compose the primary control packet and are programmable during DMA setup. The last three fields compose working control packet and are only readable by the CPU. The working control packets are used to support auto-initiation. The organization of control packets is shown in [Figure 16-5](#).

The primary control packet contains channel information such as source address, destination address, transfer count, element/frame index pointer and channel configuration. Source address, destination address and transfer count also have their respective working images. The three fields of working images compose a working control packet and are not accessible to the CPU in write access.

The first time a DMA channel is selected for a transaction, the following process occurs:

1. The primary control packet is first read by the DMA state machine.
2. Once the channel is arbitrated, the current source address, destination address and transfer count are then copied to their respective working images.
3. When the channel is serviced again by the DMA, the state machine will read both the primary control packet and the working control packet to continue the DMA transaction until the end of an entire block transfer.

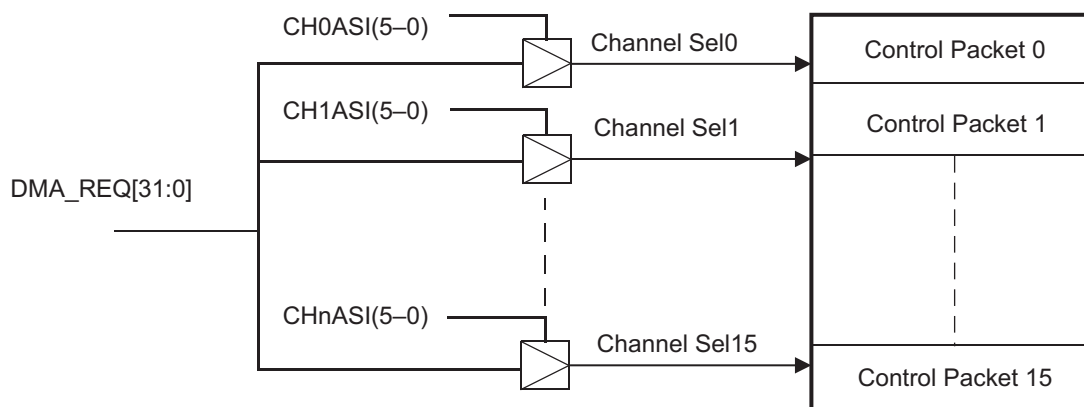
When the same channel is requested again, the state machine will start again by reading only the primary control packet and then continue the same process described above. The user software need not set up control packets again because the contents of the primary control packet were never lost. The working images of the control packets are reducing the software overhead and interaction with the DMA module to a minimum.

---

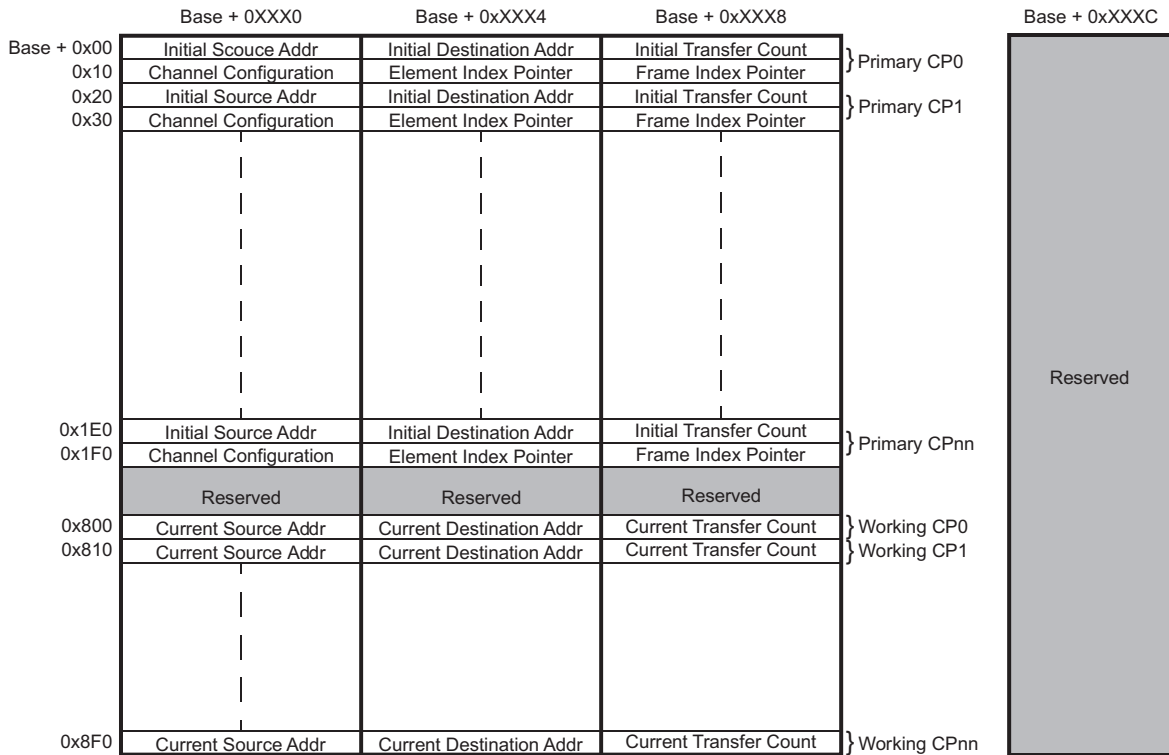
**NOTE:** Changing the contents of a channel control packet will clear the corresponding pending bit ([Section 16.3.1.2](#)) if the channel has a pending status. If the control packet of an active channel (as indicated in [Section 16.3.1.3](#)) is changed, then the channel will stop immediately at an arbitration boundary. When the same channel is triggered again, it will begin with the new control packet information.

---

**Figure 16-4. DMA Request Mapping and Control Packet Organization**



**Figure 16-5. Control Packet Organization and Memory Map**



**16.2.4.1 Initial Source Address**

This field stores the absolute 32-bit source address of the DMA transfer.

**16.2.4.2 Initial Destination Address**

This field stores the absolute 32-bit destination address of the DMA transfer.

**16.2.4.3 Initial Transfer Count**

The transfer count field is composed of two parts. The frame transfer count value and the element transfer count value. Each count value is 13 bits wide. As a Single Block transfer maximum of 512 Mbytes of data can be transferred. Element count and frame count are programmed according to the source data structure.

The total transfer size is calculated as stated below:

$$T_{sz} = E_{rsz} \cdot E_{tc} \cdot F_{tc} \tag{26}$$

where

$T_{sz}$  = Total Transfer Size

$E_{rsz}$  = Read Element Size

$E_{tc}$  = Element Transfer Count

$F_{tc}$  = Frame Transfer Count

**NOTE:** A zero element count with a non-zero frame count or a non-zero element count with a zero frame count are all considered as zero total transfer count. No DMA transaction is initiated with any of the counters set to 0.

#### 16.2.4.4 Channel Configuration Word

The channel configuration defines the following individual parameters

- Read element size
- Write element size
- Trigger type (frame or block)
- Addressing mode for source
- Addressing mode for destination
- Auto-initiation mode
- Next control packet to be triggered at control packet finish (Channel Chaining)

#### 16.2.4.5 Element/Frame Index Pointer

There are 4 index pointers that allow the creation of different types of buffers in RAM and address registers in a structured manner: an element index pointer for source and destination and a frame index pointer for source and destination.

The element index pointer for source and/or destination defines the offset to be added after each element transfer to the source and/or destination address. The frame index pointer for source and/or destination defines the offset to be added to the source and/or destination address after the element count reaches zero. The element and frame index pointers must be defined in terms of the number of bytes of offset. The DMA controller does not adjust the element/frame index number according to the element size. An index of 2 means *increment the address by 2* and not by 16 when the element size is 64 bits.

#### 16.2.4.6 Current Source Address

The current source address field contains the current working source address during a DMA transaction. The current source address is incremented during post increment addressing mode or indexing mode.

#### 16.2.4.7 Current Destination Address

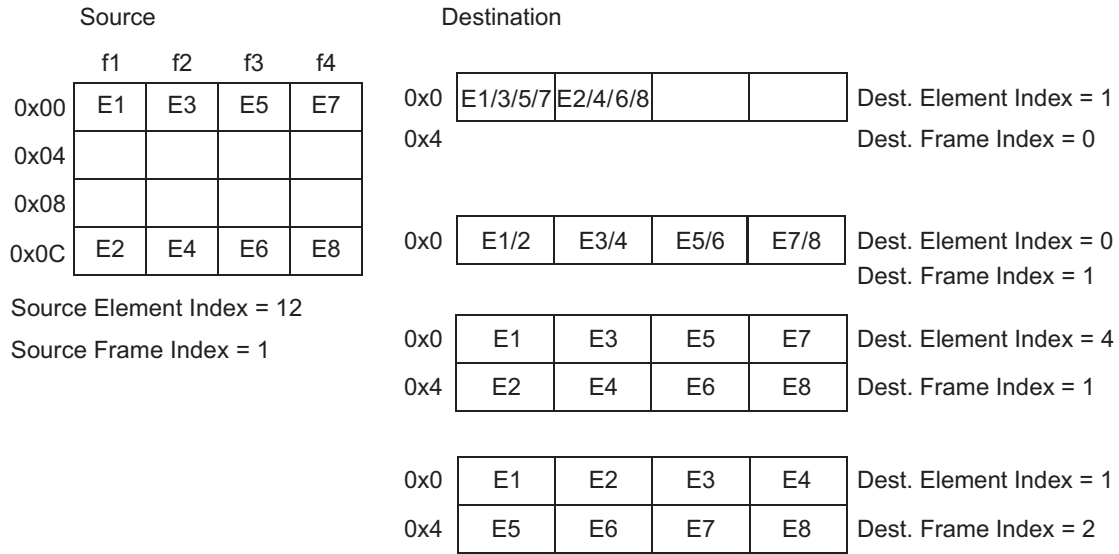
The current destination address field contains the current working destination address during a DMA transaction. The current destination address is incremented during post-increment addressing mode or indexing mode.

#### 16.2.4.8 Current Transfer Count

The current transfer count stores the remaining number of elements to be transferred in a block. It is decremented by one for each element read from the source location.

[Figure 16-6](#), [Figure 16-7](#) and [Figure 16-8](#) show some examples of DMA transfers.

**Figure 16-6. DMA Transfer Example 1**



The example assumes the following setup.

Read Element Size = 8 bit  
Write Element Size = 8 bit  
Element Count = 2  
Frame Count = 4

**Figure 16-7. DMA Indexing Example 1**

	f1	f2	f3	f4
0x0	E1	E5	E9	E13
0x10	E2	E6	E10	E14
0x20	E3	E7	E11	E15
0x30	E4	E8	E12	E16

Element Index = 16  
Frame Index = 4

This example can be applied to either source or destination indexing and assumes the following setup.

Element Size = 16 bit  
Element Count = 4  
Frame Count = 4



**Figure 16-8. DMA Indexing Example 2**

0x0	E1	E4	E7	E10	E13	E16	E19	E22
0x20								
0x40	E2	E5	E8	E11	E14	E17	E20	E23
0x60								
0x80	E3	E6	E9	E12	E15	E18	E21	E23

Element Index = 64

Frame Index = 4

This example can be applied to either source or destination indexing and assumes the following setup.

Element Size = 32 bit

Element Count = 3

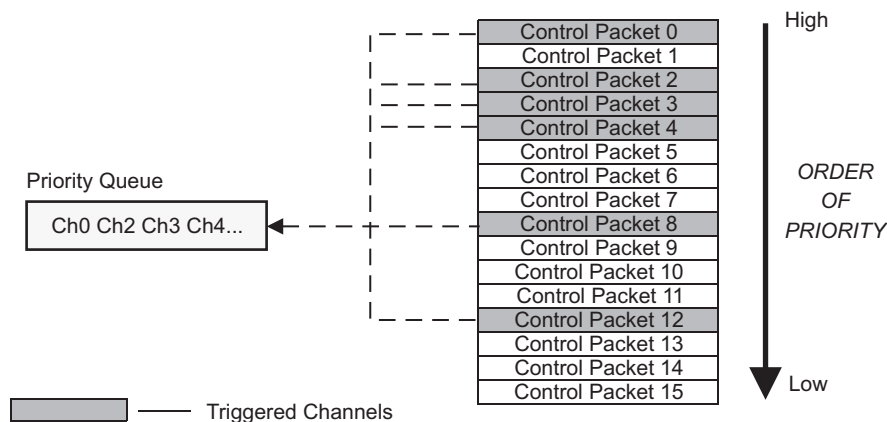
Frame Count = 8

### 16.2.5 Priority Queue

User can assign channels in to priority queues to facilitate request handling during arbitration. The port has two priority queues: a high and a low priority queue. The queue can be configured to follow a fixed or rotating priority scheme. Fixed priority is such that the lower the channel number (Figure 16-9), the higher its priority. Rotating priority is based on a round-robin scheme. Initially, the priority list is sorted according to the fixed priority scheme. Channels assigned to the high priority queue are always serviced first according to the selected priority scheme before channels in the low priority queue are serviced. Table 16-1 describes how arbitration is performed according to different priority schemes.

**NOTE:** Since the DMA controller provides the capability to map any one of the 32 hardware DMA request lines to any channel, the numerical order of the hardware DMA request does not imply any priority. The priority of each hardware DMA request is programmed and determined by software.

**Figure 16-9. Fixed Priority Scheme**



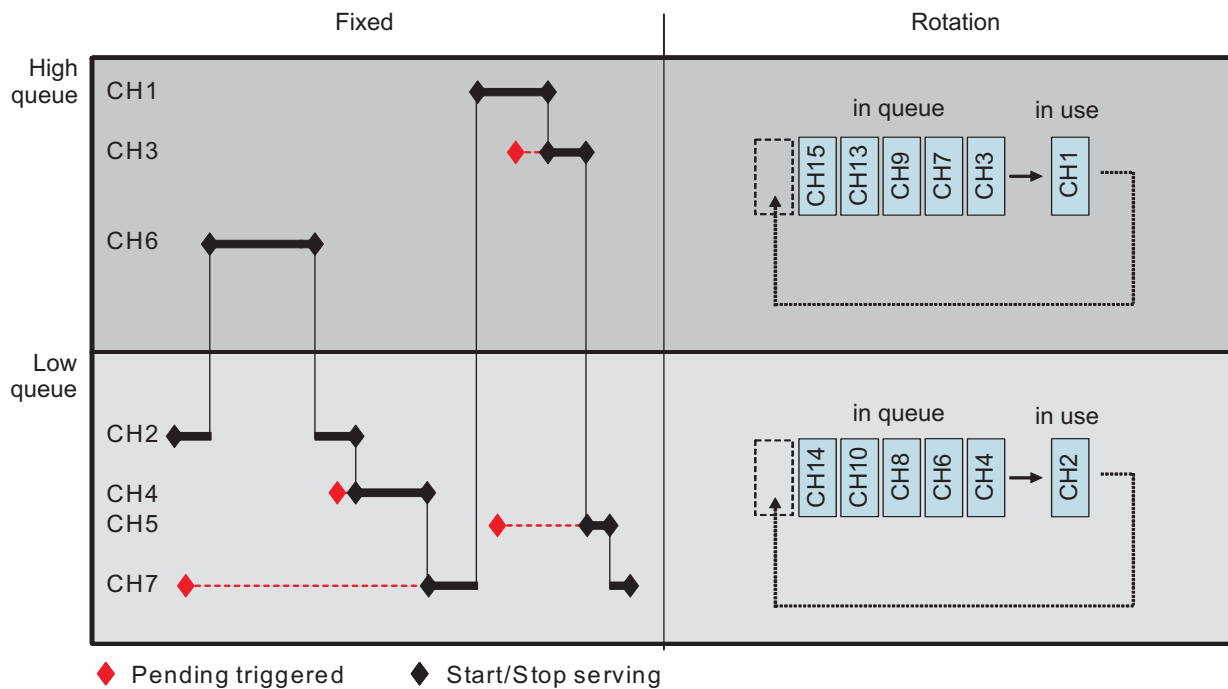
The above figure illustrates that by default Lower the channel number, higher the Priority.

**Table 16-1. Arbitration According to Priority Queues and Priority Schemes**

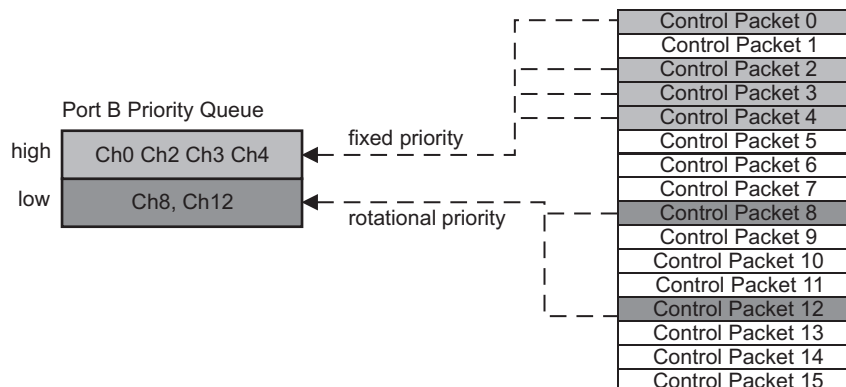
Queue	Priority Scheme	Remark
High priority	Fixed	Channels are serviced in an ascending order according to the channel number. The lower the channel number, the higher the priority. A channel will be arbitrated out whenever there is a higher pending channel. Otherwise a channel is completely serviced until its transfer count reaches zero before the next highest pending channel is serviced. When there is no pending channels left in high queue then the DMA switches to service low queue channels.
	Rotating	Channels are arbitrated by using the round-robin scheme. Arbitration is performed when the FIFO is empty. When there are no pending channels left in high queue then the DMA switches to service low queue channels.
Low priority	Fixed	Channels are serviced in an ascending order according to the channel number. The lower the channel number the higher the priority. A channel will be arbitrated out whenever there is a higher-priority pending channel. Otherwise a channel is completely serviced until its transfer count reaches zero, before the next highest pending channel is serviced. If there is a pending channel in the high-priority queue while DMA is servicing a low queue channel then DMA will switch back to service high queue channel after an arbitration boundary.
	Rotating	Channels are arbitrated by using round-robin scheme. Arbitration is performed when the FIFO is empty.

A Simple Priority Queues example in both Fixed and Rotation Scheme is shown in [Figure 16-10](#).

**Figure 16-10. Example of Priority Queues**



For optimal system performance, the high priority channels should be put in fixed arbitration scheme and low priority channels in the rotating priority scheme as illustrated in [Figure 16-11](#).

**Figure 16-11. Example Channel Assignments**

<sup>1</sup> The above figure illustrates the channel assignments in a system with 16 channels. This approach can be scaled dependent on the total channels available.

### 16.2.6 Data Packing and Unpacking

The DMA controller automatically performs the necessary data packing and unpacking when the read element size differs from the write element size. Data packing is required when the read element size is smaller than the write element size; data unpacking is required when the read element size is larger than the write element size. When the read element size is equal to the write element size, no packing is performed during read, nor is any unpacking performed during write.

Figure 16-12 shows an example of data unpacking in which the DMA is used to transfer 128 transmit data elements to the MibSPI FIFO buffer. In this example, data unpacking is required because the read element size is 64 while the write element size is 16. The DMA first performs a 64-bit read from the source into its FIFO buffer. After the 64-bit data is read into the DMA FIFO buffer, it must unpack the data into four 16-bit data elements before writing out to the destination. Therefore the DMA would need to perform four 16 bit write operations to the destination.

---

**NOTE:** In the example in Figure 16-12, to transmit data at the lower bits of the MibSPI, bits 15:0, the destination address should be incremented by a factor of 2.

---

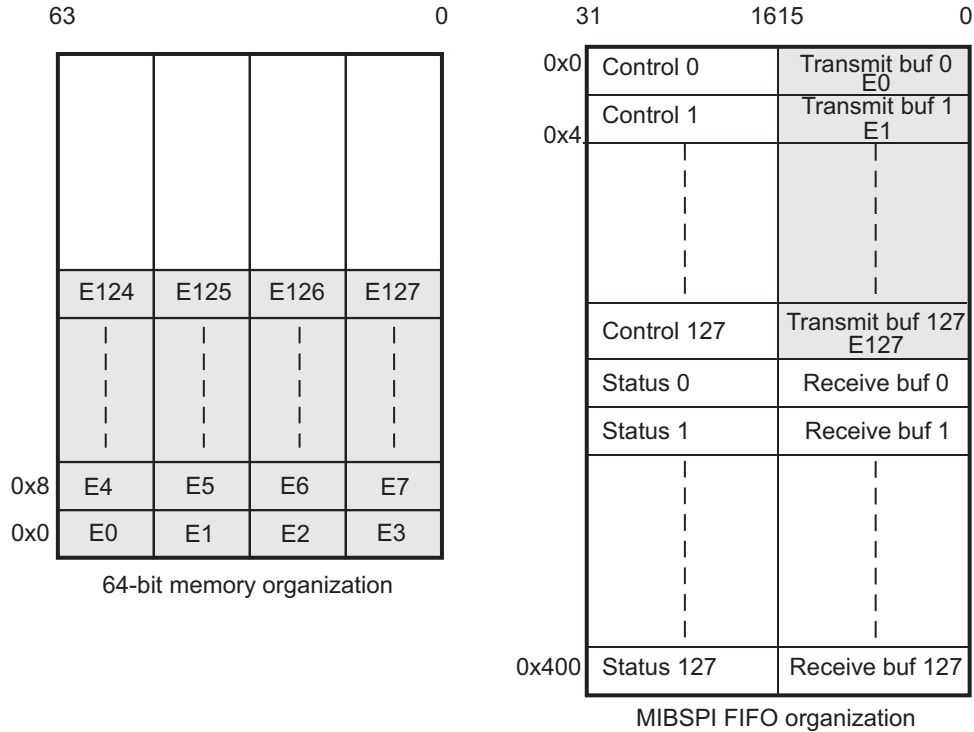
**NOTE:** 1) The element Count (Section 16.3.2.3) refers only to the read element.

2) Data unpacking does not require the DMA request. Once the DMA request is received, data from Source is moved in to FIFO and unpacking happens until the FIFO is empty.

3) DMA assumes the destination is always ready and will perform write immediately. In case of data unpacking and Constant Addressing Mode write (Section 16.3.2.4 (1 - 0) = 0) the destination data will be overwritten by next data or next data might be skipped in case the destination has overflow protection (eg., SCITD register). User should configure DMA to avoid data unpacking if the Destination is configured as Constant Addressing Mode write to avoid data loss.

---

Figure 16-12. Example of DMA Data Unpacking

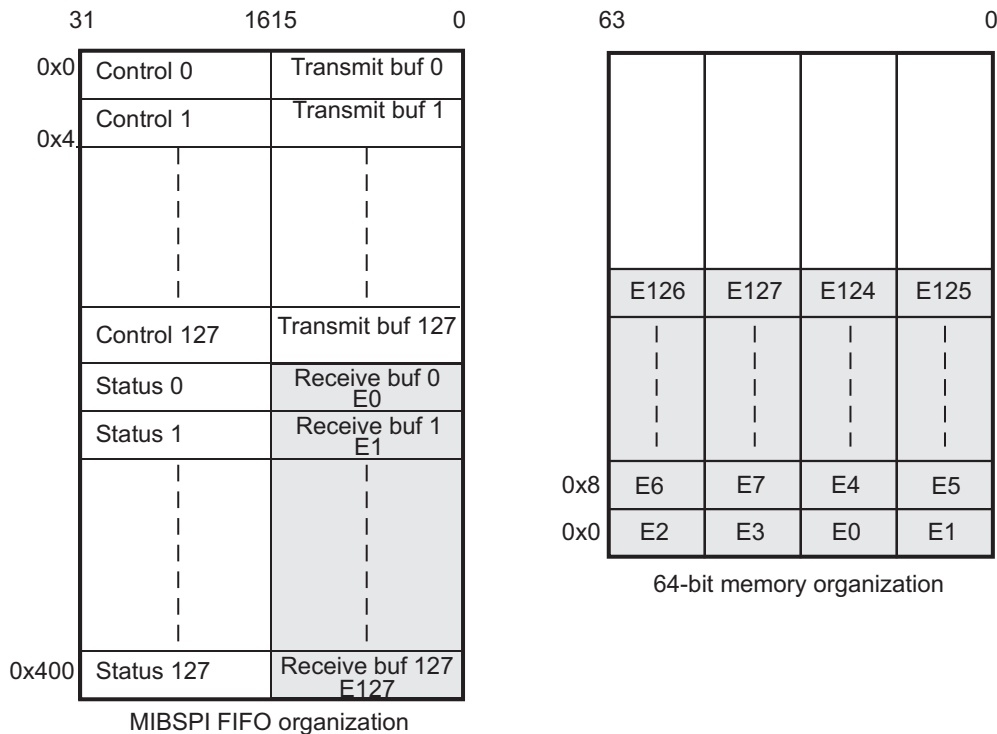


In this example, initialization of the MIBSPI FIFO is illustrated and assumes the following setup:

- Read Element Size = 64 bit
- Write Element Size = 16 bit
- Element Count = 32
- Frame Count = 1
- Source Element Index = n/a, use post increment addressing mode
- Source Frame Index = n/a, use post increment addressing mode
- Destination Element Index = 4
- Destination Frame Index = 0

When the read element size is smaller than the write element size, the DMA controller needs to perform data packing. The number of elements to pack is equal to the ratio between the write element size and read element size. In the example in Figure 16-13, the read element size is 16 bits and the write element size is 64 bits. The DMA controller would first pack the first four elements by performing four consecutive 16-bit read accesses of E0, E1, E2, and E3 into the first word of the DMA's internal FIFO. The DMA controller would then perform one single 64-bit write operation to transfer the data to the 64-bit destination memory.

Normally, the DMA controller carries out bus transactions on the bus according to the element size. For example, the DMA controller would perform a 16-bit read transaction if the read element size is programmed as 16 bits, or an 8-bit write transaction if the write element size is programmed as 8 bit. The exception is when the total transfer size as defined in Equation 26 is not a multiple of the write element size.

**Figure 16-13. Example of DMA Data Packing**

In this example, a read of the MIBSPI FIFO is illustrated and assumes the following setup:

Read Element Size = 16 bit  
 Write Element Size = 64 bit  
 Element Count = 128  
 Frame Count = 1  
 Source Element Index = 4  
 Source Frame Index = 0  
 Destination Element Index = n/a, use post increment addressing mode  
 Destination Frame Index = n/a, use post increment addressing mode

For example, if the read element size is 8 bits, the element transfer count is equal to 9, and the write element size is 64 bit. The DMA controller would first perform eight 8-bit read transactions from the source. It would then perform a 64-bit write to the destination. When the same channel wins arbitration again, the DMA controller would first perform one 8-bit read from the source, followed by one 8-bit write to the destination, even though the write element size is 64 bit.

---

**NOTE:** Since peripherals are slower, it is advised to use data packing feature with caution for reading data from peripherals. Improper use might delay servicing other pending DMA channels.

---

## 16.2.7 DMA Request

There are three ways to start a DMA transfer:

- **Software request:** The transfer will be triggered by writing to SW Channel Enable Set and Status Register (Section 16.3.1.6). The software request can trigger either a block or a frame transfer depending on what the trigger type (TTYPER) bit is set to in the Channel Control Register (Section 16.3.2.4) register.
- **Hardware request:** The DMA controller can handle up to 32 DMA Request lines. A hardware request can trigger either a frame or a block transfer dependent on the setting of TTYPER control bit of the Channel Control Register (Section 16.3.2.4) register.

- **Triggered by other control packet:** When a control packet finishes the programmed number of transfers it can trigger another channel to initiate its transfers.

Each time a DMA request is made, either one frame transfer or one block transfer can be chosen. An active DMA request signal will trigger a DMA transaction.

The DMA controller has a two-level buffer to capture HW requests per channel. When a HW request is generated and the channel is enabled, the corresponding bit in the DMA Status Register (Section 16.3.1.3) is set. The pending register acts as a first-level buffer. Typically, a peripheral acting as a source of a transfer would initiate another request after its data registers have been read out by DMA, even though that data has not been completely transferred to the destination. If a second HW request is generated by the peripheral, the DMA controller has an extra request buffer to capture this second request and service it after the first request is complete.

---

**NOTE:** The DMA cannot capture more than three requests if its request buffers are already full. If any request occur during this moment DMA will discard it.

---

The DMA controller also supports a mix of hardware and software requests on the same channel. If a software request is generated, the corresponding bit in the Channel Pending Register (Section 16.3.1.2) is set accordingly. If the pending request is not completely serviced by the DMA and a hardware request is generated by a peripheral onto the same channel, the DMA will capture and recognize this hardware request into its request buffer.

---

**NOTE:** The DMA controller cannot recognize two software requests on the same channel if the first software request is still pending. If such request occur DMA will discard it. Therefore the user S/W should check the pending register before issue a new software request.

---

### 16.2.8 Auto-Initiation

When Auto-initiation Mode (AIM) bit of Channel Control Register (Section 16.3.2.4) register is enabled for a channel and the channel is triggered by a software request for a block transfer, the channel will restart again using the same channel information stored at the respective control packet after one block transfer is completed. In the case of Hardware Request the channel needs to be retriggered each time after a block is complete even if auto-initiation is enabled.

### 16.2.9 Interrupts

Each channel can be configured to generate interrupts on several transfer conditions:

- Frame transfer complete (FTC) interrupt: an interrupt is issued after the last element of a frame has been transferred.
- Last frame transfer started (LFS) interrupt: an interrupt is issued before the first element of the last frame of a block transfer has started.
- First half of block complete (HBC) interrupt: an interrupt is issued if more than half of the block is transferred.
  - If the number of frames  $n$  is odd, then the HBC interrupt is generated at the beginning of the frame when  $(n-1) / 2$  number of frames are left in the block.
  - If the number of frames  $n$  is even, then the HBC interrupt is generated at the beginning of the frame after  $n/2$  number of frames are left in the block.
- Block transfer complete (BTC) interrupt: an interrupt is issued after the last element of the last frame has been transferred.
- Bus error (BER) interrupt: an interrupt is issued when DMA detects an error on the bus. The BER interrupt is connected to the ESM module.
- Memory Protection Unit error (MPU): an interrupt is issued when the DMA detects that the access falls outside of a memory region programmed in the MPU registers of the DMA. The MPU interrupt is connected to the ESM module.
- Parity error (PAR): an interrupt is issued when the DMA detects a parity error when reading one of the

control packets. The PAR interrupt is connected to the ESM module.

The DMA outputs 5 interrupt lines for control packet handling, a parity interrupt and a memory protection interrupt (Figure 16-14). Each type of transfer interrupt condition is grouped together. For example, all block-transfer complete interrupts that are routed to a port are combined (OR'ed). The channel that caused the interrupt is given in the corresponding interrupt channel offset register. Priority between interrupts among the same interrupt type is resolved by a fixed priority scheme. Priority between different interrupt types is resolved in the Vector Interrupt Manager. Figure 16-15 explains the Frame Transfer Complete Interrupt structure in detail.

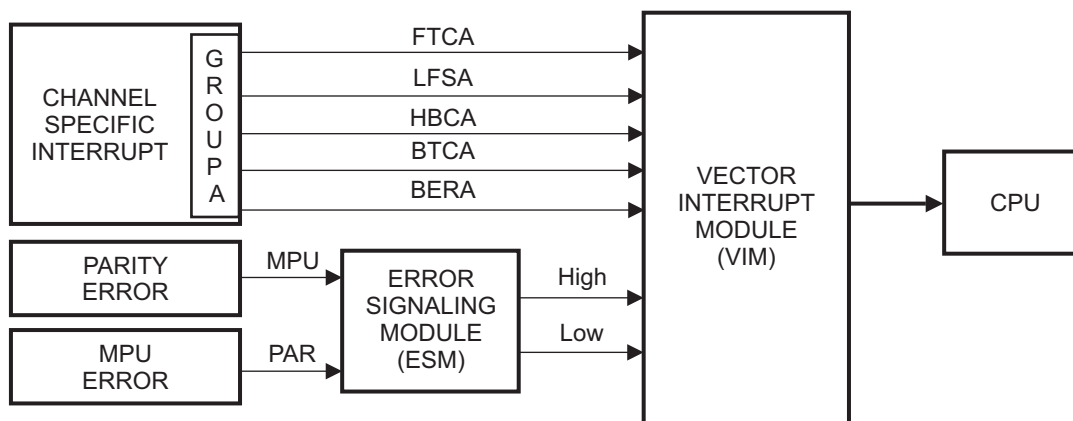
**NOTE:** Each Channel Specific interrupts in DMA module are routed towards Group A or B to support two different CPUs individually. For devices with Single CPU or Dual CPU where both CPUs are running same code in delayed lock-step as safety feature:

Group A - Interrupts (FTC, LFS, HBC, BTC and BER) are routed to the ARM CPU.

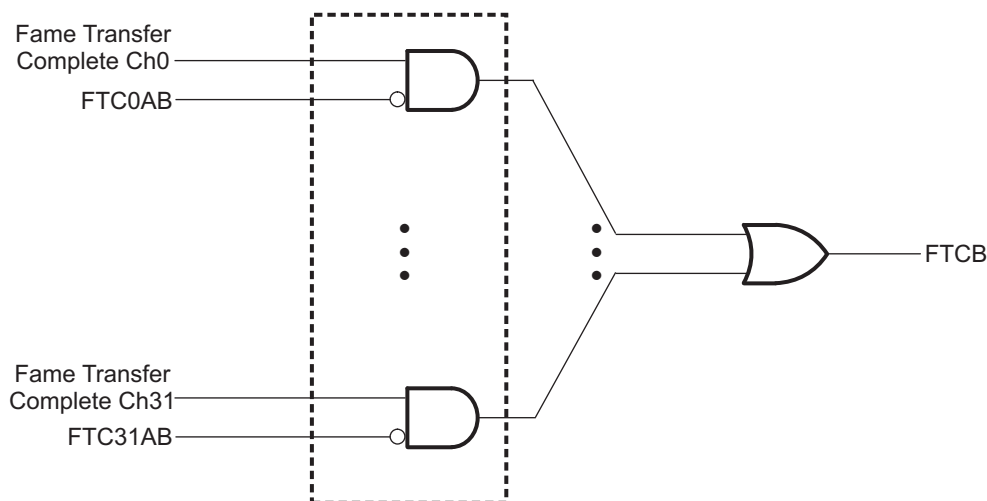
Group B - Interrupts (FTC, LFS, HBC, BTC and BER) are not routed out.

User software should configure only Group A interrupts.

**Figure 16-14. DMA Interrupts**



**Figure 16-15. Detailed Interrupt Structure (Frame Transfer Complete Path)**



1 This figure is applicable for HBC, LFS, BTC, and BER interrupt

### 16.2.10 Debugging

The DMA supports four different behaviors in suspend mode. These behaviors can be configured by the user as per the application requirement.

- Immediate stop at a DMA channel arbitration boundary. Please refer to [Table 16-2](#) and [Table 16-3](#) for arbitration boundary definition.
- Finish current frame transfer and continue after suspend ends.
- Finish current block transfer and continue after suspend ends.
- Ignore the suspend. The DMA continues to be operational as in functional mode when debug mode is active.

When the DMA controller enters suspend mode, it continues to sample incoming hardware DMA requests, but the Channel Pending Register ([Section 16.3.1.2](#)) is frozen from being updated. After the suspend ends, all new requests that were received during suspend mode are reflected in the Channel Pending Register ([Section 16.3.1.2](#)).

Except when the DMA controller is configured to ignore suspend mode, no channel arbitration is performed during suspend mode. The current channel under which suspend mode was entered will finish its entire frame or block-transfer after suspend mode ends, depending how the debug option was chosen.

To facilitate debugging, a Watch Point Register ([Section 16.3.1.50](#)) and a Watch Mask Register ([Section 16.3.1.51](#)) are used. The watch point register together with the watch mask register can be configured to watch for a unique address or a range of addresses. When the condition to watch is true, the DMA freezes its state and generates a debug request signal to the host CPU so the state of the DMA can be examined.

### 16.2.11 Power Management

The DMA offers two power-management modes: run and sleep. In run mode, the DMA is fully operational.

The sleep mode shuts down the DMA if no pending channels are waiting to be serviced. If a DMA request is received or a software request is generated by the user software, then the DMA wakes up immediately.

The sleep mode may be used to optimize the DMA module power consumption.

When the system module issues a global low power mode request, the DMA will respond to the system module with an acknowledge if no DMA requests are pending.

---

**NOTE:** When the DMA is in global low power mode, the clock is stopped and therefore it cannot detect any DMA request. The device must be woken up before a peripheral can generate a DMA request.

---

### 16.2.12 FIFO Buffer

DMA FIFO is 4 levels deep and 64-bit wide (can hold upto 4 x 64-bits of data). They are used for Data packing and unpacking.

The DMA FIFO has two states:

- EMPTY : The FIFO contains no data.
- FULL : The FIFO is filled or the element count has reached zero; the read operation has to be stopped.

DMA channels can only be switched when the FIFO is empty. This also implies that arbitration between channels is done when the FIFO is empty.

The FIFO buffer may be bypassed through the use of the bypass feature in the port control register; see Port Control Register ([Section 16.3.1.47](#)) for register details. Writing 1 to this bit limits the FIFO depth to the size of one element. That means after one element is read the write out to the destination will start. This feature is particularly useful to minimize switching latency in-between channels. When bypass mode is enabled, the DMA will perform minimal bus cycles on AHB bus. In addition, the bypass feature allows arbitration between channels that can be carried out at a source element granularity.



However, it has to be considered that while in bypass mode, the DMA controller does not make optimal use of the bus bandwidth. Since the read and write element sizes can be different, then the number of read and write transactions will be different. [Table 16-2](#) and [Table 16-3](#) show a comparison between the number of read and write transactions performed by the DMA controller from one channel to another before arbitration in non-bypass and bypass mode.

**Table 16-2. Maximum Number of DMA Transactions per Channel in Non-Bypass Mode**

	Write Element Size	8 bit		16 bit		32 bit		64 bit	
Read Element Size	8 bit	4 read	4 write	4 read	2 write	4 read	1 write	8 read	1 write
	16 bit	2 read	4 write	4 read	4 write	4 read	2 write	4 read	1 write
	32 bit	1 read	4 write	2 read	4 write	4 read	4 write	4 read	2 write
	64 bit	1 read	8 write	1 read	4 write	2 read	4 write	4 read	4 write

**Table 16-3. Maximum Number of DMA Transactions per Channel in Bypass Mode**

	Write Element Size	8 bit		16 bit		32 bit		64 bit	
Read Element Size	8 bit	1 read	1 write	2 read	1 write	4 read	1 write	8 read	1 write
	16 bit	1 read	2 write	1 read	1 write	2 read	1 write	4 read	1 write
	32 bit	1 read	4 write	1 read	2 write	1 read	1 write	2 read	1 write
	64 bit	1 read	8 write	1 read	4 write	1 read	2 write	1 read	1 write

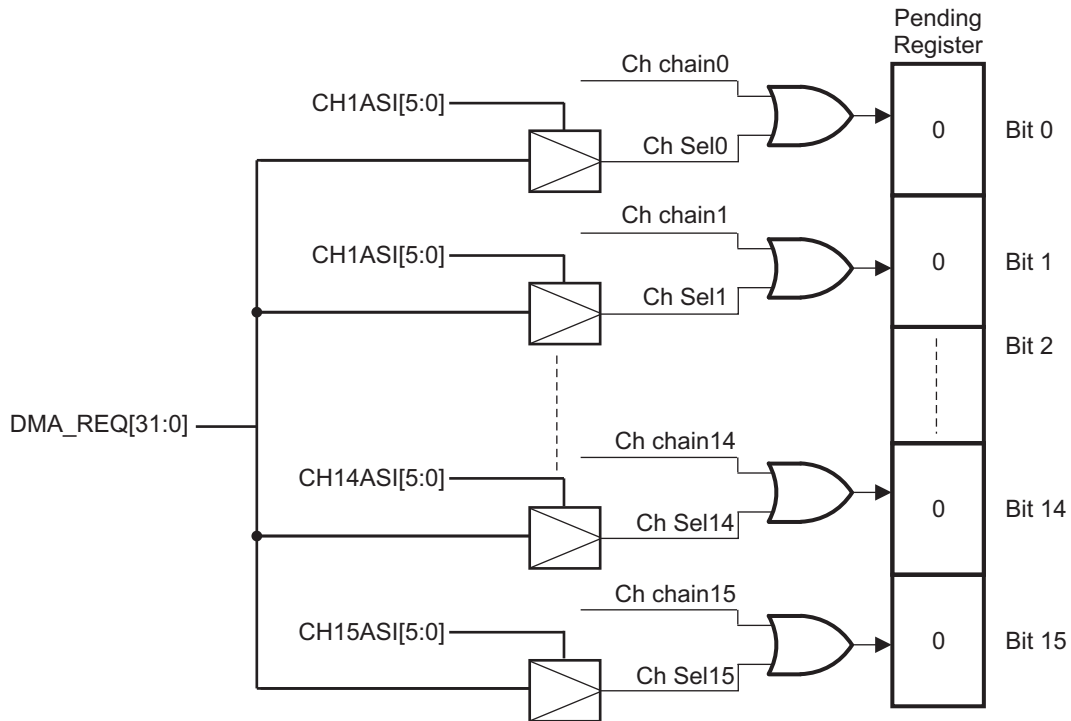
### 16.2.13 Channel Chaining

Channel chaining is used to trigger a single or multiple channels with out an external DMA request. This is possible by chaining one control packet to other. Chain[5:0] field of the Channel Control Register ([Section 16.3.2.4](#)) is used to program the chaining control packet. Chained control packets follow arbitration rules within the pending register. For example if CH1, CH2, CH4, CH5 are triggered together and CH3 is chained with CH1. The order of channels serviced in spite of chaining will be CH1 -> CH2 -> CH3 -> CH4 -> CH5.

In order to setup up channel chain feature, the Channel Control Register ([Section 16.3.2.4](#)) register needs to be enabled for all chained channels before triggering first DMA request.

[Figure 16-16](#) illustrates how internally chained request is generated after completing the required transfers and stored in pending register. In this example CH1 is Chained to CH0. When CH0 is triggered CH1 is captured as pending in the Channel Pending Register ([Section 16.3.1.2](#)) even when it is not triggered.

Figure 16-16. Example of Channel Chaining



### 16.2.14 Memory Protection

The DMA controller is capable of access to the full address range of the device. The protection mechanism allows the protection of up to four memory regions to restrict accesses to those address ranges. This will allow the application to protect critical application data from unintentionally being accessed by the DMA controller.

#### 16.2.14.1 Protection Mechanism

The memory protection mechanism consists of the access privilege for a given memory region, the start and end address for the region, and notification of an access violation for the protected region.

Each region to be protected is configured by software by writing the start address and end address for each region into the DMA Memory Protection Registers, DMAMPxS and DMAMPxE. The definition of these registers can be found starting at [Section 16.3.1.60](#). Any region in the valid address space can be protected from inappropriate accesses.

The access privileges can be set to one of four permission settings as shown below:

- Full access
- Read only access
- Write only access
- No access

The permissions for a given region are selected by writing the appropriate values in the DMA Memory Protection Control Register ([Section 16.3.1.60](#)).

---

**NOTE:** If the regions defined by the start and end addresses overlap, the region defined first in the register space determines the access privilege. For example, if region 0 and region 1 overlap, the access permissions defined for region 0 will take precedence since region 0 registers are before region 1.

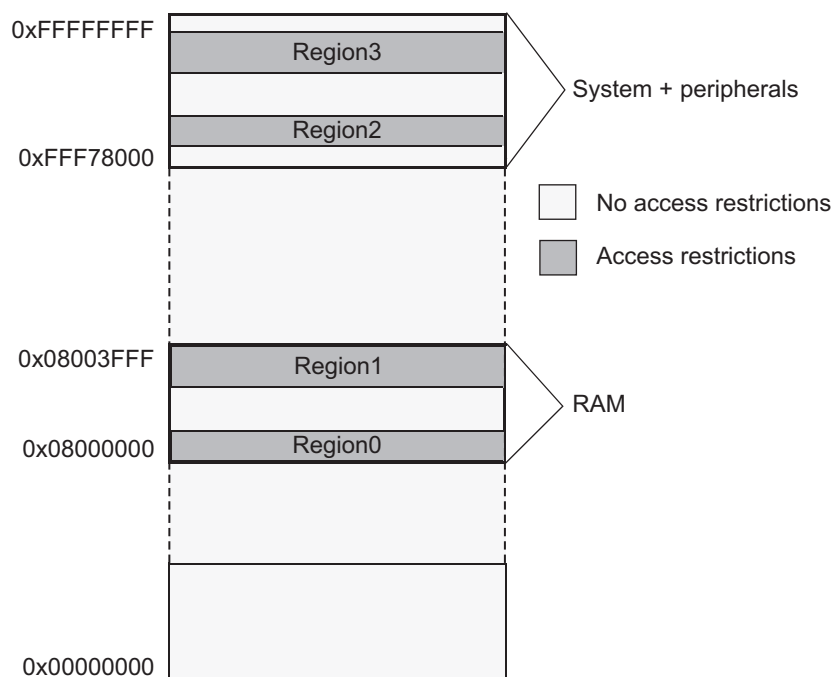
---

In a case where a memory protection violation occurs, a flag will be set and an interrupt will be generated,

if interrupts are enabled. The DMA Memory Protection Status Register ([Section 16.3.1.61](#)) contains the status flags for the memory protection mechanism, and the DMA Memory Protection Control Register ([Section 16.3.1.60](#)) contains the interrupt enable bits. Upon detection of the memory protection violation, the DMA Channel that caused the violation will be stopped and the next available DMA channel will be serviced.

[Figure 16-17](#) illustrates a protection mechanism.

**Figure 16-17. Example of Protection Mechanism**



### 16.2.15 Parity Checking

Parity checking is implemented using parity on a per-byte basis for DMA Control Packets in the RAM. Checking for even or odd parity can be programmed by a 4-bit key located in the system module that controls the parity configuration on a global basis. This ensures that all modules using parity are acting in the same manner. The default setup after reset is odd parity.

In addition, parity checking can be enabled and disabled within the module by a 4-bit key. The key is located in the Parity Control Register ([Section 16.3.1.58](#)).

During a read access, regardless if it was read by the DMA state machine or another master (CPU), the parity is calculated based on the data read from the RAM and compared with the good parity value stored in the parity bits. If any word fails the parity check, then a parity error interrupt is generated. The address that generated the error is detected and is captured for host system debugging in the DMA Parity Error Address Register ([Section 16.3.1.59](#)). The address is frozen from being updated until it is read by the bus master.

Additional error handling is dependent on the requestor.

- DMA reading from a control packet RAM: The transmission requested by DMA request will not take place.
- CPU reading from the control packet RAM: The data will be retrieved by the CPU and a parity error interrupt will be generated.

In both cases, the control packet will be left active or the DMA will be switched off dependent on the ERRA bit in the Parity Control Register ([Section 16.3.1.58](#)).

### 16.2.16 Parity Testing

The parity RAM is accessible to allow manually inserting faults so that the parity checking feature can be tested. Test mode is entered by asserting the TEST bit within the [Section 16.3.1.58](#) register. Once the bit is set, the parity bits are mapped to the control packet RAM starting address A00h.

---

**NOTE:** When in test mode, no parity checking will be done when reading from parity memory, but parity checking will be performed on the normal memory.

---

Each byte in Control Packet RAM has its own parity bit in the Control Packet Parity RAM as shown in [Table 16-4](#), [Table 16-5](#) and [Table 16-6](#). P0 is the parity bit for byte 0, P1 is the parity bit for byte 1 and so on.

Each byte in the control packet RAM has its own parity bit in the control packet parity RAM as shown in [Table 16-4](#) and [Table 16-5](#).

**Table 16-4. Control Packet RAM**

Bit	31	24	23	16	15	8	7	0
Word 0	Byte 0		Byte 1		Byte 2		Byte 3	
Word 1	Byte 4		Byte 5		Byte 6		Byte 7	
Word 2	Byte 8		Byte 9		Byte 10		Byte 11	
Word 3	Byte 12		Byte 13		Byte 14		Byte 15	

**Table 16-5. Control Packet RAM**

Bit	127	96	95	64	63	32	31	0
	Word 3		Word 2		Word 1		Word 0	

**Table 16-6. Parity RAM**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0

### 16.2.17 Initializing RAM with Parity

After power up, the RAM content including the parity bit cannot be guaranteed. To avoid parity failures when reading RAM, the RAM has to be initialized. The RAM can be initialized by writing known values into it. When the known value is written, the corresponding parity bit will be automatically calculated and updated.

Another possibility to initialize the memory is to set a 4-bit key in the system module. This key triggers the automatic initialization of all RAMs on the microcontroller. The RAM will be initialized to 0. Depending on the even/odd parity selection, the parity bit will be calculated accordingly.

To allow for parity calculation during initialization, the parity functionality has to be enabled as discussed in [Section 16.2.15](#).

## 16.3 Control Registers and Control Packets

This section details the DMA control registers and memory map, summarized in [Table 16-7](#). [Table 16-8](#) summarizes the control packets. Each register begins on a word boundary. All registers and control packets are accessible in 8, 16, and 32 bit. The base address for the control registers is FFFF F000h.

---

**NOTE:** The register definitions are given for a full DMA module configuration (32 channels, 64 requests, 2 Ports, Dual CPU support). Writes and Reads of bits pertaining to features not included in the DMA implementation as defined in the device specific datasheet are possible without error; however, they will have no affect on device operation.

---



---

**NOTE:** Control registers and control packets have two different base addresses.

---

**Table 16-7. DMA Control Registers**

Offset	Acronym	Register Description	Section
00	GCTRL	Global Control Register	<a href="#">Section 16.3.1.1</a>
04h	PEND	Channel Pending Register	<a href="#">Section 16.3.1.2</a>
0Ch	DMASTAT	DMA Status Register	<a href="#">Section 16.3.1.3</a>
14h	HWCHENAS	HW Channel Enable Set and Status Register	<a href="#">Section 16.3.1.4</a>
1Ch	HWCHENAR	HW Channel Enable Reset and Status Register	<a href="#">Section 16.3.1.5</a>
24h	SWCHENAS	SW Channel Enable Set and Status Register	<a href="#">Section 16.3.1.6</a>
2Ch	SWCHENAR	SW Channel Enable Reset and Status Register	<a href="#">Section 16.3.1.7</a>
34h	CHPRIOS	Channel Priority Set Register	<a href="#">Section 16.3.1.8</a>
3Ch	CHPRIOR	Channel Priority Reset Register	<a href="#">Section 16.3.1.9</a>
44h	GCHIENAS	Global Channel Interrupt Enable Set Register	<a href="#">Section 16.3.1.10</a>
4Ch	GCHIENAR	Global Channel Interrupt Enable Reset Register	<a href="#">Section 16.3.1.11</a>
54h	DREQASIO	DMA Request Assignment Register 0	<a href="#">Section 16.3.1.12</a>
58h	DREQASI1	DMA Request Assignment Register 1	<a href="#">Section 16.3.1.13</a>
5Ch	DREQASI2	DMA Request Assignment Register 2	<a href="#">Section 16.3.1.14</a>
60h	DREQASI3	DMA Request Assignment Register 3	<a href="#">Section 16.3.1.15</a>
94h	PAR0	Port Assignment Register 0	<a href="#">Section 16.3.1.16</a>
98h	PAR1	Port Assignment Register 1	<a href="#">Section 16.3.1.17</a>
B4h	FTCMAP	FTC Interrupt Mapping Register	<a href="#">Section 16.3.1.18</a>
BCh	LFSMAP	LFS Interrupt Mapping Register	<a href="#">Section 16.3.1.19</a>
C4h	HBCMAP	HBC Interrupt Mapping Register	<a href="#">Section 16.3.1.20</a>
CCh	BTCMAP	BTC Interrupt Mapping Register	<a href="#">Section 16.3.1.21</a>
D4h	BERMAP	BER Interrupt Mapping Register	<a href="#">Section 16.3.1.22</a>
DCh	FTCINTENAS	FTC Interrupt Enable Set	<a href="#">Section 16.3.1.23</a>
E4h	FTCINTENAR	FTC Interrupt Enable Reset	<a href="#">Section 16.3.1.24</a>
ECh	LFSINTENAS	LFS Interrupt Enable Set	<a href="#">Section 16.3.1.25</a>
F4h	LFSINTENAR	LFS Interrupt Enable Reset	<a href="#">Section 16.3.1.26</a>
FCh	HBCINTENAS	HBC Interrupt Enable Set	<a href="#">Section 16.3.1.27</a>
104h	HBCINTENAR	HBC Interrupt Enable Reset	<a href="#">Section 16.3.1.28</a>
10Ch	BTCINTENAS	BTC Interrupt Enable Set	<a href="#">Section 16.3.1.29</a>
114h	BTCINTENAR	BTC Interrupt Enable Reset	<a href="#">Section 16.3.1.30</a>
11Ch	GINTFLAG	Global Interrupt Flag Register	<a href="#">Section 16.3.1.31</a>
124h	FTCFLAG	FTC Interrupt Flag Register	<a href="#">Section 16.3.1.32</a>
12Ch	LFSFLAG	LFS Interrupt Flag Register	<a href="#">Section 16.3.1.33</a>
134h	HBCFLAG	HBC Interrupt Flag Register	<a href="#">Section 16.3.1.34</a>
13Ch	BTCFLAG	BTC Interrupt Flag Register	<a href="#">Section 16.3.1.35</a>

**Table 16-7. DMA Control Registers (continued)**

<b>Offset</b>	<b>Acronym</b>	<b>Register Description</b>	<b>Section</b>
144h	BERFLAG	BER Interrupt Flag Register	<a href="#">Section 16.3.1.36</a>
14Ch	FTCAOFFSET	FTCA Interrupt Channel Offset Register	<a href="#">Section 16.3.1.37</a>
150h	LFSAOFFSET	LFSA Interrupt Channel Offset Register	<a href="#">Section 16.3.1.38</a>
154h	HBCAOFFSET	HBCA Interrupt Channel Offset Register	<a href="#">Section 16.3.1.39</a>
158h	BTCAOFFSET	BTCA Interrupt Channel Offset Register	<a href="#">Section 16.3.1.40</a>
15Ch	BERAOFFSET	BERA Interrupt Channel Offset Register	<a href="#">Section 16.3.1.41</a>
160h	FTCBOFFSET	FTCB Interrupt Channel Offset Register	<a href="#">Section 16.3.1.42</a>
164h	LFSBOFFSET	LFSB Interrupt Channel Offset Register	<a href="#">Section 16.3.1.43</a>
168h	HBCBOFFSET	HBCB Interrupt Channel Offset Register	<a href="#">Section 16.3.1.44</a>
16Ch	BTCBOFFSET	BTCB Interrupt Channel Offset Register	<a href="#">Section 16.3.1.45</a>
170h	BERBOFFSET	BERB Interrupt Channel Offset Register	<a href="#">Section 16.3.1.46</a>
178h	PTCRL	Port Control Register	<a href="#">Section 16.3.1.47</a>
17Ch	RTCTRL	RAM Test Control Register	<a href="#">Section 16.3.1.48</a>
180h	DCTRL	Debug Control	<a href="#">Section 16.3.1.49</a>
184h	WPR	Watch Point Register	<a href="#">Section 16.3.1.50</a>
188h	WMR	Watch Mask Register	<a href="#">Section 16.3.1.51</a>
18Ch	PAACSADDR	Port A Active Channel Source Address Register	<a href="#">Section 16.3.1.52</a>
190h	PAACDADDR	Port A Active Channel Destination Address Register	<a href="#">Section 16.3.1.53</a>
194h	PAACTC	Port A Active Channel Transfer Count Register	<a href="#">Section 16.3.1.54</a>
198h	PBACSADDR	Port B Active Channel Source Address Register	<a href="#">Section 16.3.1.55</a>
19Ch	PBACDADDR	Port B Active Channel Destination Address Register	<a href="#">Section 16.3.1.56</a>
1A0h	PBACTC	Port B Active Channel Transfer Count Register	<a href="#">Section 16.3.1.57</a>
1A8h	DMAPCR	Parity Control Register	<a href="#">Section 16.3.1.58</a>
1ACh	DMAPAR	DMA Parity Error Address Register	<a href="#">Section 16.3.1.59</a>
1B0h	DMAMPCTRL	DMA Memory Protection Control Register	<a href="#">Section 16.3.1.60</a>
1B4h	DMAMPST	DMA Memory Protection Status Register	<a href="#">Section 16.3.1.61</a>
1B8h	DMAMPROS	DMA Memory Protection Region 0 Start Address Register	<a href="#">Section 16.3.1.62</a>
1BCh	DMAMPROE	DMA Memory Protection Region 0 End Address Register	<a href="#">Section 16.3.1.63</a>
1C0h	DMAMPR1S	DMA Memory Protection Region 1 Start Address Register	<a href="#">Section 16.3.1.64</a>
1C4h	DMAMPR1E	DMA Memory Protection Region 1 End Address Register	<a href="#">Section 16.3.1.65</a>
1C8h	DMAMPR2S	DMA Memory Protection Region 2 Start Address Register	<a href="#">Section 16.3.1.66</a>
1CCh	DMAMPR2E	DMA Memory Protection Region 2 End Address Register	<a href="#">Section 16.3.1.67</a>
1D0h	DMAMPR3S	DMA Memory Protection Region 3 Start Address Register	<a href="#">Section 16.3.1.68</a>
1D4h	DMAMPR3E	DMA Memory Protection Region 3 End Address Register	<a href="#">Section 16.3.1.69</a>

**Table 16-8. Control Packet Memory Map**

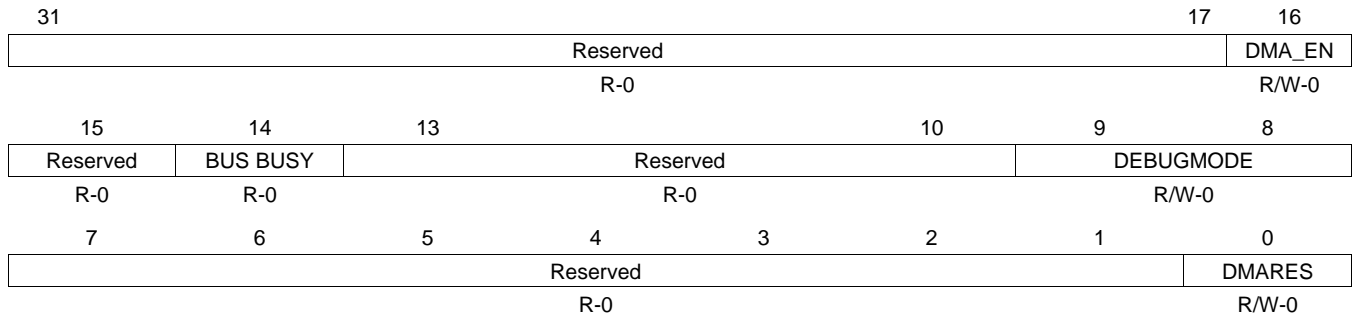
Offset	Acronym	Register Description	Section
<b>Primary Control Packet 0</b>			
00	ISADDR	Initial Source Address Register	<a href="#">Section 16.3.2.1</a>
04h	IDADDR	Initial Destination Address Register	<a href="#">Section 16.3.2.2</a>
08h	ITCOUNT	Initial Transfer Count Register	<a href="#">Section 16.3.2.3</a>
10h	CHCTRL	Channel Control Register	<a href="#">Section 16.3.2.4</a>
14h	EIOFF	Element Index Offset Register	<a href="#">Section 16.3.2.5</a>
18h	FIOFF	Frame Index Offset Register	<a href="#">Section 16.3.2.6</a>
<b>Working Control Packet 0</b>			
800h	CSADDR	Current Source Address Register	<a href="#">Section 16.3.2.7</a>
804h	CDADDR	Current Destination Address Register	<a href="#">Section 16.3.2.8</a>
808h	CTCOUNT	Current Transfer Count Register	<a href="#">Section 16.3.2.9</a>

### 16.3.1 Global Configuration Registers

These registers control the overall behavior of the DMA controller.

#### 16.3.1.1 Global Control Register (GCTRL)

**Figure 16-18. Global Control Register (GCTRL) [offset = 00]**



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

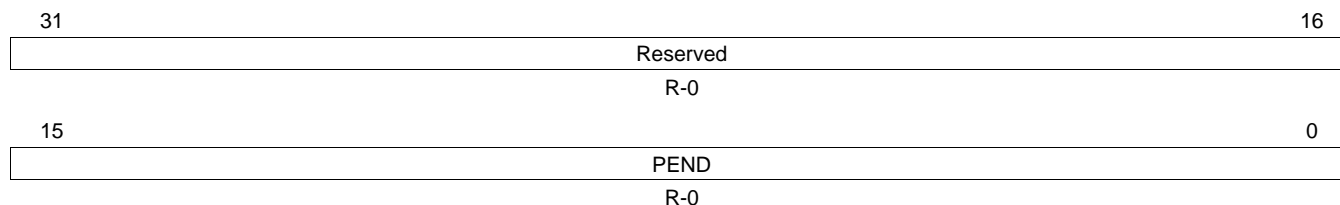
**Table 16-9. Global Control Register (GCTRL) Field Descriptions**

Bit	Field	Value	Description
31-17	Reserved	0	Read returns 0. Writes have no effect.
16	DMA_EN	0 1	<p>DMA enable bit. The configuration registers and channel control packets should be setup first before DMA_EN bit is set to one to prevent state machines from carrying out bus transactions. If DMA_EN bit is cleared in the middle of an bus transaction, the state machine will stop at an arbitration boundary.</p> <p>0 The DMA is disabled. 1 The DMA is enabled.</p>
15	Reserved	0	Read returns 0. Writes have no effect.
14	BUS BUSY	0 1	<p>This bit indicates status of DMA external AHB bus status.</p> <p>0 DMA's external bus is not busy in data transfers. 1 DMA's external bus is busy in data transfers.</p>
13-10	Reserved	0	Read returns 0. Writes have no effect.
9-8	DEBUG MODE	0 1h 2h 3h	<p>0 Ignore suspend. 1h Finish current block transfer. 2h Finish current frame transfer. 3h Immediately stop at an DMA arbitration boundary and continue after suspend.</p>
7-1	Reserved	0	Read returns 0. Writes have no effect.
0	DMA RES	0 1	<p>DMA software reset</p> <p>0 Software reset is disabled. 1 The DMA state machine and all control registers are in software reset. Control packets are not reset when DMA software reset is active.</p> <p><b>Note:</b> In the event a DMA slave does not respond, the DMA module will respond to the software reset upon reaching an arbitration boundary.</p>



### 16.3.1.2 Channel Pending Register (PEND)

**Figure 16-19. Channel Pending Register (PEND) [offset = 04h]**



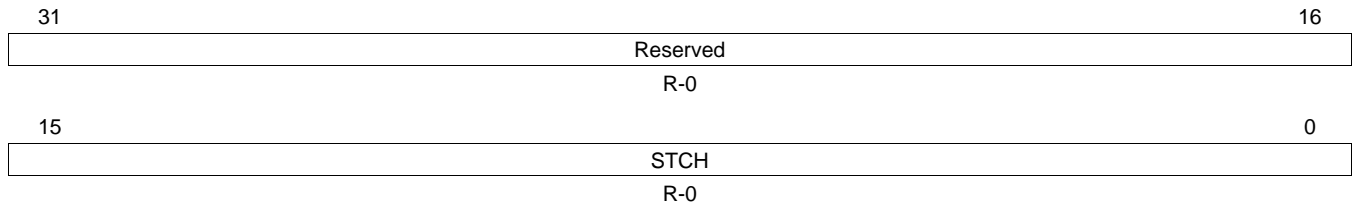
LEGEND: R = Read only; -n = value after reset

**Table 16-10. Channel Pending Register (PEND) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	PEND	0	Channel pending register. Reading from PEND gives the channel pending information no matter if the channel was initiated by SW or HW. Once set, it remains set even if the corresponding channel is disabled via HWCHEANA or SWCHEANA. The pending bit is automatically cleared for the following conditions: <ul style="list-style-type: none"> <li>• At the end of a frame or a block transfer depending on how the channel is triggered as programmed in the TTYPE bit field of CHCTRL.</li> <li>• The control packet is modified after the pending bit is set.</li> <li>• An AHB bus error occurs.</li> </ul>
		1	The corresponding channel is pending and is waiting for service.

**16.3.1.3 DMA Status Register (DMASTAT)**

**Figure 16-20. DMA Status Register (DMASTAT) [offset = 0Ch]**



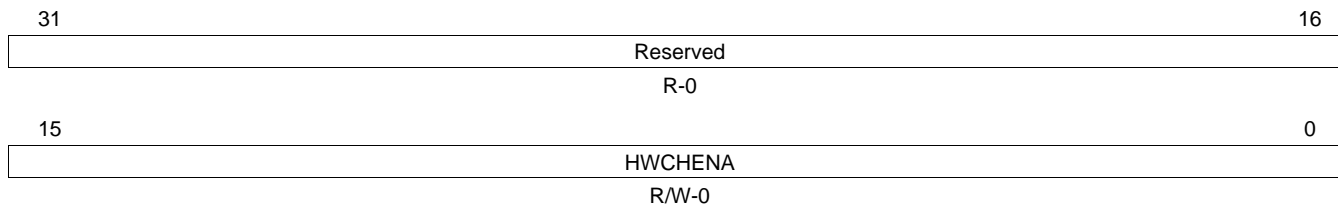
LEGEND: R = Read only; -n = value after reset

**Table 16-11. DMA Status Register (DMASTAT) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	STCH	0	The channel is inactive.
		1	The channel is active; that is, the channel is currently in the DMA's execution queue.
			Note: The status of a channel currently in DMA's execution queue remains active even if emulation mode is entered or DMA is disabled via DMA_EN bit.

### 16.3.1.4 HW Channel Enable Set and Status Register (HWCHENAS)

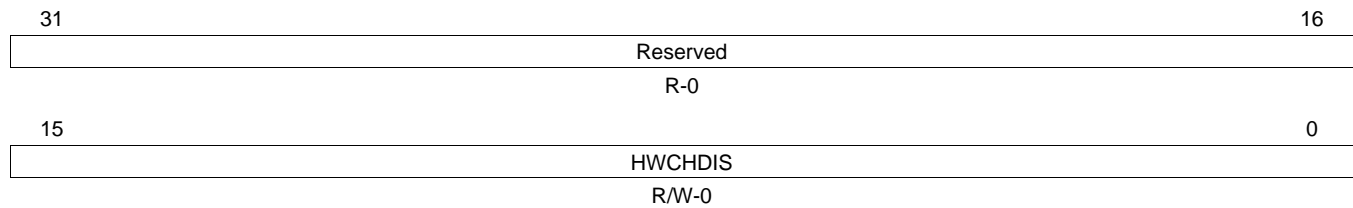
**Figure 16-21. HW Channel Enable Set and Status Register (HWCHENAS) [offset = 14h]**



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

**Table 16-12. HW Channel Enable Set and Status Register (HWCHENAS) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	HWCHENA		<p>Hardware channel enable bit. An active hardware DMA request cannot initiate a DMA transfer unless the corresponding hardware enable bit is set.</p> <p>The corresponding hardware enable bit is cleared automatically for the following conditions:</p> <ul style="list-style-type: none"> <li>• At the end of a block transfer if the auto-initiation bit AIM (see CHCTRL) is not active.</li> <li>• If an AHB bus error is detected for an active channel.</li> </ul> <p>Reading from HWCHENAS gives the status (enabled/disabled) of channels 0 through 15.</p>
		0	The corresponding channel is disabled for hardware triggering.
		1	The corresponding channel is enabled for hardware triggering.

**16.3.1.5 HW Channel Enable Reset and Status Register (HWCHENAR)**
**Figure 16-22. HW Channel Enable Reset and Status Register (HWCHENAR) [offset = 1Ch]**


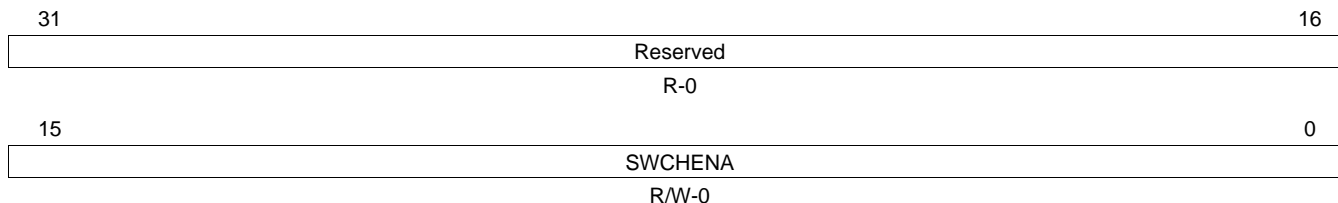
LEGEND: R = Read only; -n = value after reset

**Table 16-13. HW Channel Enable Reset and Status Register (HWCHENAR) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	HWCHDIS	0	HW channel disable bit. Read: The corresponding channel is disabled for HW triggering. Write: A write of zero to this bit has no effect.
		1	Read: The corresponding channel is enabled for HW triggering. Write: The corresponding channel is disabled.

### 16.3.1.6 SW Channel Enable Set and Status Register (SWCHENAS)

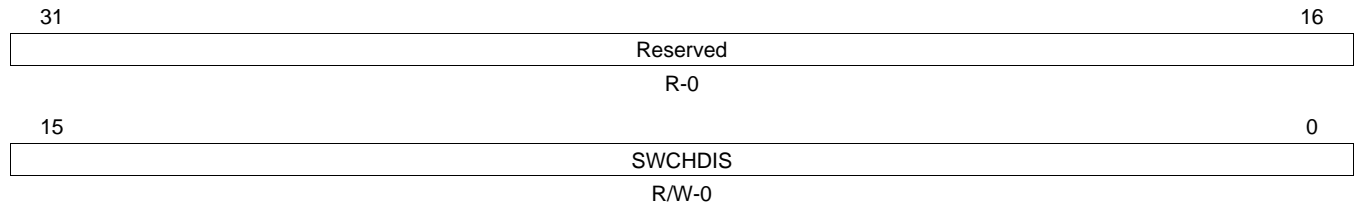
**Figure 16-23. SW Channel Enable Set and Status Register (SWCHENAS) [offset = 24h]**



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

**Table 16-14. SW Channel Enable Set and Status Register (SWCHENAS) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	SWCHENA		<p>SW channel enable bit. Writing a one to a bit triggers a SW request on the corresponding channel to start a DMA transaction. The corresponding bit is automatically cleared by the following conditions.</p> <ul style="list-style-type: none"> <li>• The corresponding bit is cleared after one frame transfer if the TTYPE bit in Channel Control Register (CHCTRL) is programmed for frame transfer.</li> <li>• The corresponding bit is cleared after one block transfer if the corresponding TTYPE bit is programmed for block transfer and the auto-initiation bit is not enabled.</li> <li>• The control packet is modified after the pending bit is set.</li> <li>• The corresponding bit is cleared after one block transfer when TTYPE bit is programmed for blocks transfer and if the corresponding bit in HW channel enable register (HWCHENAS) is enabled. When a channel is enabled for both HW and SW, the state machine will initiate transfers based on the SW first. After one block transfer is complete, the corresponding bit in the SWCHENA register is then cleared. The same channel is serviced again by a HW DMA request.</li> <li>• The corresponding bit is cleared if an AHB bus error is detected.</li> </ul> <p>Reading from SWCHENAS gives the status (enabled/disabled) of channels 0 through 15.</p> <p style="margin-left: 20px;">0     The corresponding channel is not triggered by SW request.</p> <p style="margin-left: 20px;">1     The corresponding channel is triggered by SW request.</p>

**16.3.1.7 SW Channel Enable Reset and Status Register (SWCHENAR)**
**Figure 16-24. SW Channel Enable Reset and Status Register (SWCHENAR) [offset = 2Ch]**


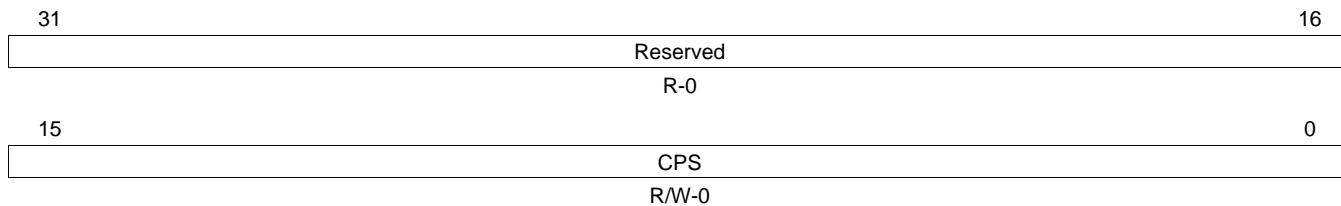
LEGEND: R = Read only; -n = value after reset

**Table 16-15. SW Channel Enable Reset and Status Register (SWCHENAR) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	SWCHDIS	0	SW channel disable bit. Read: The corresponding channel was not triggered by SW. Write: Writing a zero to this bit has no effect.
		1	Read: The corresponding channel was triggered by SW. Write: The corresponding channel is disabled.

### 16.3.1.8 Channel Priority Set Register (CHPRIOS)

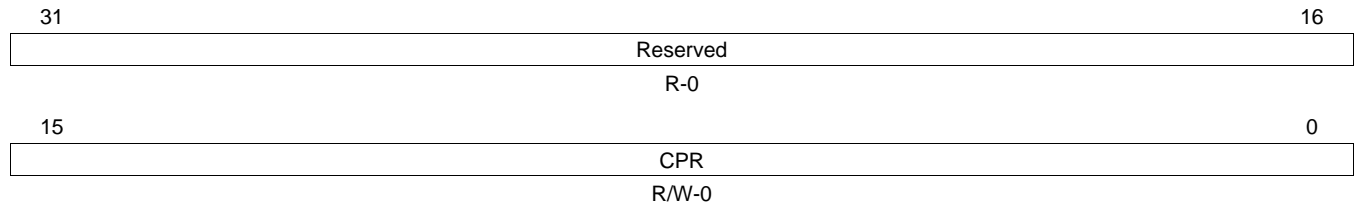
**Figure 16-25. Channel Priority Set Register (CHPRIOS) [offset = 34h]**



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

**Table 16-16. Channel Priority Set Register (CHPRIOS) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	CPS	0	Channel priority set bit. Writing a one to a bit assigns the corresponding channel to the high priority queue. Read: The corresponding channel is assigned to the low priority queue. Write: Writing a zero to this bit has no effect.
		1	Read and write: The corresponding channel is assigned to high priority queue.

**16.3.1.9 Channel Priority Reset Register (CHPRIOR)**
**Figure 16-26. Channel Priority Reset Register (CHPRIOR) [offset = 3Ch]**


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

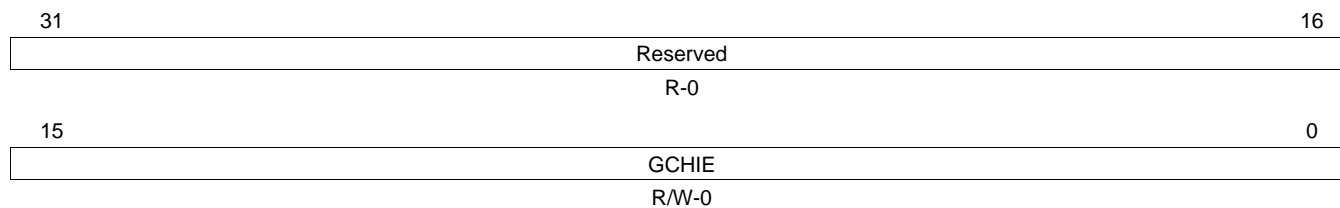
**Table 16-17. Channel Priority Reset Register (CHPRIOR) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	CPR	0	Channel priority reset bit. Writing a one to a bit assigns the according channel to the low priority queue. Read: The corresponding channel is assigned to the low priority queue. Write: Writing a zero to this bit has no effect.
		1	Read: The corresponding channel is assigned to the high priority queue. Write: The corresponding channel is assigned to the low priority queue.



### 16.3.1.10 Global Channel Interrupt Enable Set Register (GCHIENAS)

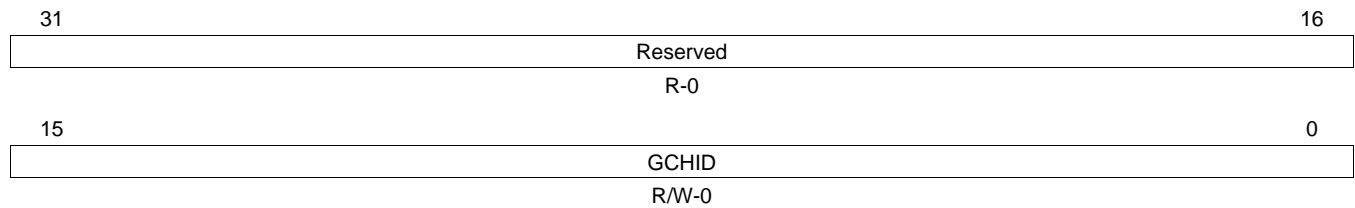
**Figure 16-27. Global Channel Interrupt Enable Set Register (GCHIENAS) [offset = 44h]**



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

**Table 16-18. Global Channel Interrupt Enable Set Register (GCHIENAS) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	GCHIE	0	Global channel interrupt enable bit. Read: The corresponding channel is disabled for interrupt. Write: Writing a zero to this bit has no effect.
		1	Read and write: The corresponding channel is enabled for interrupt.

**16.3.1.11 Global Channel Interrupt Enable Reset Register (GCHIENAR)**
**Figure 16-28. Global Channel Interrupt Enable Reset Register (GCHIENAR) [offset = 4Ch]**


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

**Table 16-19. Global Channel Interrupt Enable Reset Register (GCHIENAR) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	GCHID	0	Global channel interrupt disable bit. Read: The corresponding channel is disabled for interrupt. Write: Writing a zero to this bit has no effect.
		1	Read: The corresponding channel is enabled for interrupt. Write: The corresponding channel is disabled for interrupt.

### 16.3.1.12 DMA Request Assignment Register 0 (DREQASI0)

**Figure 16-29. DMA Request Assignment Register 0 (DREQASI0) [offset = 54h]**

31	30	29	24	23	22	21	16
Reserved	CH0ASI			Reserved	CH1ASI		
R-0	R/W-0			R-0	R/W-1h		
15	14	13	8	7	6	5	0
Reserved	CH2ASI			Reserved	CH3ASI		
R-0	R/W-2h			R-0	R/W-3h		

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

**Table 16-20. DMA Request Assignment Register 0 (DREQASI0) Field Descriptions**

Bit	Field	Value	Description
31-30	Reserved	0	Read returns 0. Writes have no effect.
	CH0ASI	0 ... 1Fh	Channel 0 assignment. This bit field chooses the DMA request assignment for channel 0. DMA request line 0 triggers channel 0. ... DMA request line 31 triggers channel 0.
23-22	Reserved	0	Read returns 0. Writes have no effect.
	CH1ASI	0 ... 1Fh	Channel 1 assignment. This bit field chooses the DMA request assignment for channel 1. DMA request line 0 triggers channel 1. ... DMA request line 31 triggers channel 1.
15-14	Reserved	0	Read returns 0. Writes have no effect.
13-8	CH2ASI	0 ... 1Fh	Channel 2 assignment. This bit field chooses the DMA request assignment for channel 2. DMA request line 0 triggers channel 2. ... DMA request line 31 triggers channel 2.
7-6	Reserved	0	Read returns 0. Writes have no effect.
5-0	CH3ASI	0 ... 1Fh	Channel 3 assignment. This bit field chooses the DMA request assignment for channel 3. DMA request line 0 triggers channel 3. ... DMA request line 31 triggers channel 3.

**16.3.1.13 DMA Request Assignment Register 1 (DREQASI1)**
**Figure 16-30. DMA Request Assignment Register 1 (DREQASI1) [offset = 58h]**

31	30	29	24	23	22	21	16
Reserved	CH4ASI			Reserved	CH5ASI		
R-0	R/W-4h			R-0	R/W-5h		
15	14	13	8	7	6	5	0
Reserved	CH6ASI			Reserved	CH7ASI		
R-0	R/W-6h			R-0	R/W-7h		

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

**Table 16-21. DMA Request Assignment Register 1 (DREQASI1) Field Descriptions**

Bit	Field	Value	Description
31-30	Reserved	0	Read returns 0. Writes have no effect.
29-24	CH4ASI	0 ... 1Fh	Channel 4 assignment. This bit field chooses the DMA request assignment for channel 4. DMA request line 0 triggers channel 4. ... DMA request line 31 triggers channel 4.
23-22	Reserved	0	Read returns 0. Writes have no effect.
21-26	CH5ASI	0 ... 1Fh	Channel 5 assignment. This bit field chooses the DMA request assignment for channel 5. DMA request line 0 triggers channel 5. ... DMA request line 31 triggers channel 5.
15-14	Reserved	0	Read returns 0. Writes have no effect.
13-8	CH6ASI	0 ... 1Fh	Channel 6 assignment. This bit field chooses the DMA request assignment for channel 6. DMA request line 0 triggers channel 6. ... DMA request line 31 triggers channel 6.
7-6	Reserved	0	Read returns 0. Writes have no effect.
5-0	CH7ASI	0 ... 1Fh	Channel 7 assignment. This bit field chooses the DMA request assignment for channel 7. DMA request line 0 triggers channel 7. ... DMA request line 31 triggers channel 7.

### 16.3.1.14 DMA Request Assignment Register 2 (DREQASI2)

**Figure 16-31. DMA Request Assignment Register 2 (DREQASI2) [offset = 5Ch]**

31	30	29	24	23	22	21	16
Reserved		CH8ASI			Reserved		CH9ASI
R-0		R/W-8h			R-0		R/W-9h
15	14	13	8	7	6	5	0
Reserved		CH10ASI			Reserved		CH11ASI
R-0		R/W-Ah			R-0		R/W-Bh

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

**Table 16-22. DMA Request Assignment Register 2 (DREQASI2) Field Descriptions**

Bit	Field	Value	Description
31-30	Reserved	0	Read returns 0. Writes have no effect.
29-24	CH8ASI	0 ... 1Fh	Channel 8 assignment. This bit field chooses the DMA request assignment for channel 8. DMA request line 0 triggers channel 8. ... DMA request line 31 triggers channel 8.
23-22	Reserved	0	Read returns 0. Writes have no effect.
21-16	CH9ASI	0 ... 1Fh	Channel 9 assignment. This bit field chooses the DMA request assignment for channel 9. DMA request line 0 triggers channel 9. ... DMA request line 31 triggers channel 9.
15-14	Reserved	0	Read returns 0. Writes have no effect.
13-8	CH10ASI	0 ... 1Fh	Channel 10 assignment. This bit field chooses the DMA request assignment for channel 10. DMA request line 0 triggers channel 10. ... DMA request line 31 triggers channel 10.
7-6	Reserved	0	Read returns 0. Writes have no effect.
5-0	CH11ASI	0 ... 1Fh	Channel 11 assignment. This bit field chooses the DMA request assignment for channel 11. DMA request line 0 triggers channel 11. ... DMA request line 31 triggers channel 11.

**16.3.1.15 DMA Request Assignment Register 3 (DREQASI3)**
**Figure 16-32. DMA Request Assignment Register 3 (DREQASI3) [offset = 60h]**

31	30	29	24	23	22	21	16
Reserved	CH12ASI			Reserved	CH13ASI		
R-0	R/W-Ch			R-0	R/W-Dh		
15	14	13	8	7	6	5	0
Reserved	CH14ASI			Reserved	CH15ASI		
R-0	R/W-Eh			R-0	R/W-Fh		

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

**Table 16-23. DMA Request Assignment Register 3 (DREQASI3) Field Descriptions**

Bit	Field	Value	Description
31-30	Reserved	0	Read returns 0. Writes have no effect.
29-24	CH12ASI	0 ... 1Fh	Channel 12 assignment. This bit field chooses the DMA request assignment for channel 12. DMA request line 0 triggers channel 12. ... DMA request line 31 triggers channel 12.
23-22	Reserved	0	Read returns 0. Writes have no effect.
21-16	CH13ASI	0 ... 1Fh	Channel 13 assignment. This bit field chooses the DMA request assignment for channel 13. DMA request line 0 triggers channel 13. ... DMA request line 31 triggers channel 13.
15-14	Reserved	0	Read returns 0. Writes have no effect.
13-8	CH14ASI	0 ... 1Fh	Channel 14 assignment. This bit field chooses the DMA request assignment for channel 14. DMA request line 0 triggers channel 14. ... DMA request line 31 triggers channel 14.
7-6	Reserved	0	Read returns 0. Writes have no effect.
5-0	CH15ASI	0 ... 1Fh	Channel 15 assignment. This bit field chooses the DMA request assignment for channel 15. DMA request line 0 triggers channel 15. ... DMA request line 31 triggers channel 15.

### 16.3.1.16 Port Assignment Register 0 (PAR0)

**Figure 16-33. Port Assignment Register 0 (PAR0) [offset = 94h]**

31	30	28	27	26	24	23	22	20	19	18	16
Res.	CH0PA		Res.	CH1PA		Res.	CH2PA		Res.	CH3PA	
R-0	R/W-0		R-0	R/W-0		R-0	R/W-0		R-0	R/W-0	
15	14	12	11	10	8	7	6	4	3	2	0
Res.	CH4PA		Res.	CH5PA		Res.	CH6PA		Res.	CH7PA	
R-0	R/W-0		R-0	R/W-0		R-0	R/W-0		R-0	R/W-0	

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

**Table 16-24. Port Assignment Register 0 (PAR0) Field Descriptions**

Bit	Field	Value	Description
31	Reserved	0	Read returns 0. Writes have no effect.
30-28	CH0PA	0 1h 2h 3h 1xx 0xx	These bit fields determine to which port channel 0 is assigned. Port A1/2 combined, A1 read/A2 write Port A1/2 combined, A2 read/A1 write Port A1 only Port A2 only Port B Reserved
27	Reserved	0	Read returns 0. Writes have no effect.
26-24	CH1PA	0 1h 2h 3h 1xx 0xx	These bit fields determine to which port channel 1 is assigned. Port A1/2 combined, A1 read/A2 write Port A1/2 combined, A2 read/A1 write Port A1 only Port A2 only Port B Reserved
23	Reserved	0	Read returns 0. Writes have no effect.
22-20	CH2PA	0 1h 2h 3h 1xx 0xx	These bit fields determine to which port channel 2 is assigned. Port A1/2 combined, A1 read/A2 write Port A1/2 combined, A2 read/A1 write Port A1 only Port A2 only Port B Reserved
19	Reserved	0	Read returns 0. Writes have no effect.
18-16	CH3PA	0 1h 2h 3h 1xx 0xx	These bit fields determine to which port channel 3 is assigned. Port A1/2 combined, A1 read/A2 write Port A1/2 combined, A2 read/A1 write Port A1 only Port A2 only Port B Reserved
15	Reserved	0	Read returns 0. Writes have no effect.

**Table 16-24. Port Assignment Register 0 (PAR0) Field Descriptions (continued)**

Bit	Field	Value	Description
14-12	CH4PA		These bit fields determine to which port channel 4 is assigned.
		0	Port A1/2 combined, A1 read/A2 write
		1h	Port A1/2 combined, A2 read/A1 write
		2h	Port A1 only
		3h	Port A2 only
		1xx	Port B
		0xx	Reserved
11	Reserved	0	Read returns 0. Writes have no effect.
10-8	CH5PA		These bit fields determine to which port channel 5 is assigned.
		0	Port A1/2 combined, A1 read/A2 write
		1h	Port A1/2 combined, A2 read/A1 write
		2h	Port A1 only
		3h	Port A2 only
		1xx	Port B
		0xx	Reserved
7	Reserved	0	Read returns 0. Writes have no effect.
6-4	CH6PA		These bit fields determine to which port channel 6 is assigned.
		0	Port A1/2 combined, A1 read/A2 write
		1h	Port A1/2 combined, A2 read/A1 write
		2h	Port A1 only
		3h	Port A2 only
		1xx	Port B
		0xx	Reserved
3	Reserved	0	Read returns 0. Writes have no effect.
2-0	CH7PA		These bit fields determine to which port channel 7 is assigned.
		0	Port A1/2 combined, A1 read/A2 write
		1h	Port A1/2 combined, A2 read/A1 write
		2h	Port A1 only
		3h	Port A2 only
		1xx	Port B
		0xx	Reserved



**16.3.1.17 Port Assignment Register 1 (PAR1)****Figure 16-34. Port Assignment Register 1 (PAR1) [offset = 98h]**

31	30	28	27	26	24	23	22	20	19	18	16
Res.	CH8PA		Res.	CH9PA		Res.	CH10PA		Res.	CH11PA	
R-0	R/W-0		R-0	R/W-0		R-0	R/W-0		R-0	R/W-0	
15	14	12	11	10	8	7	6	4	3	2	0
Res.	CH12PA		Res.	CH13PA		Res.	CH14PA		Res.	CH15PA	
R-0	R/W-0		R-0	R/W-0		R-0	R/W-0		R-0	R/W-0	

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

**Table 16-25. Port Assignment Register 1 (PAR1) Field Descriptions**

Bit	Field	Value	Description
31	Reserved	0	Read returns 0. Writes have no effect.
30-28	CH8PA	0 1h 2h 3h 1xx 0xx	These bit fields determine to which port channel 8 is assigned. Port A1/2 combined, A1 read/A2 write Port A1/2 combined, A2 read/A1 write Port A1 only Port A2 only Port B Reserved
27	Reserved	0	Read returns 0. Writes have no effect.
26-24	CH9PA	0 1h 2h 3h 1xx 0xx	These bit fields determine to which port channel 9 is assigned. Port A1/2 combined, A1 read/A2 write Port A1/2 combined, A2 read/A1 write Port A1 only Port A2 only Port B Reserved
23	Reserved	0	Read returns 0. Writes have no effect.
22-20	CH10PA	0 1h 2h 3h 1xx 0xx	These bit fields determine to which port channel 10 is assigned. Port A1/2 combined, A1 read/A2 write Port A1/2 combined, A2 read/A1 write Port A1 only Port A2 only Port B Reserved
19	Reserved	0	Read returns 0. Writes have no effect.
18-16	CH11PA	0 1h 2h 3h 1xx 0xx	These bit fields determine to which port channel 11 is assigned. Port A1/2 combined, A1 read/A2 write Port A1/2 combined, A2 read/A1 write Port A1 only Port A2 only Port B Reserved
15	Reserved	0	Read returns 0. Writes have no effect.

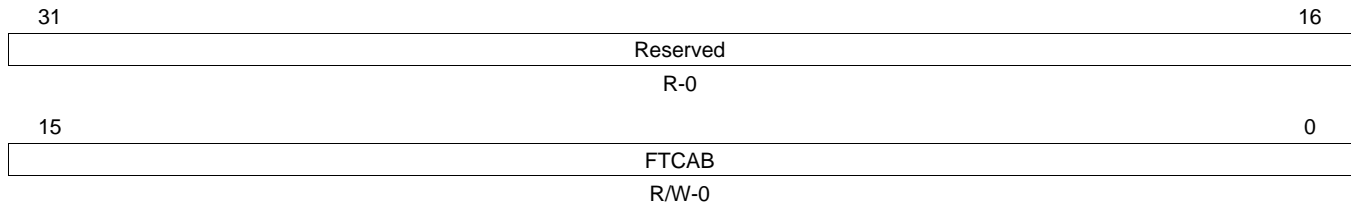
**Table 16-25. Port Assignment Register 1 (PAR1) Field Descriptions (continued)**

Bit	Field	Value	Description
14-12	CH12PA		These bit fields determine to which port channel 12 is assigned.
		0	Port A1/2 combined, A1 read/A2 write
		1h	Port A1/2 combined, A2 read/A1 write
		2h	Port A1 only
		3h	Port A2 only
		1xx	Port B
		0xx	Reserved
11	Reserved	0	Read returns 0. Writes have no effect.
10-8	CH13PA		These bit fields determine to which port channel 13 is assigned.
		0	Port A1/2 combined, A1 read/A2 write
		1h	Port A1/2 combined, A2 read/A1 write
		2h	Port A1 only
		3h	Port A2 only
		1xx	Port B
		0xx	Reserved
7	Reserved	0	Read returns 0. Writes have no effect.
6-4	CH14PA		These bit fields determine to which port channel 14 is assigned.
		0	Port A1/2 combined, A1 read/A2 write
		1h	Port A1/2 combined, A2 read/A1 write
		2h	Port A1 only
		3h	Port A2 only
		1xx	Port B
		0xx	Reserved
3	Reserved	0	Read returns 0. Writes have no effect.
2-0	CH15PA		These bit fields determine to which port channel 15 is assigned.
		0	Port A1/2 combined, A1 read/A2 write
		1h	Port A1/2 combined, A2 read/A1 write
		2h	Port A1 only
		3h	Port A2 only
		1xx	Port B
		0xx	Reserved

### 16.3.1.18 FTC Interrupt Mapping Register (FTCMAP)

**NOTE:** On this device Group B Interrupts are not Implemented hence user software should configure only Group A interrupts.

**Figure 16-35. FTC Interrupt Mapping Register (FTCMAP) [offset = B4h]**



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

**Table 16-26. FTC Interrupt Mapping Register (FTCMAP) Field Descriptions**

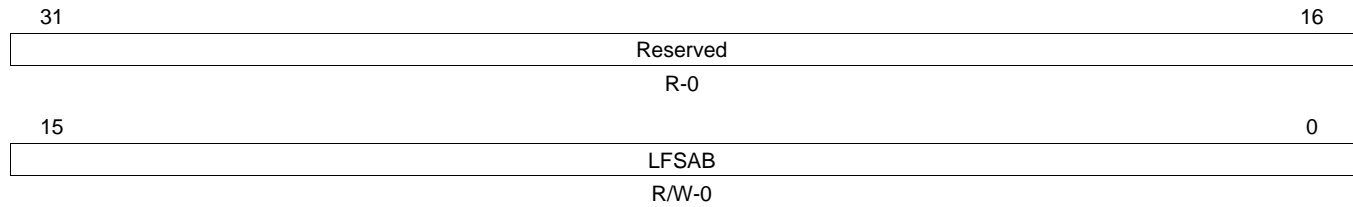
Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	FTCAB	0	Frame transfer complete (FTC) interrupt to Group A or Group B. The FTC interrupt of the corresponding channel is routed to Group A.
		1	The FTC interrupt of the corresponding channel is routed to Group B.

**16.3.1.19 LFS Interrupt Mapping Register (LFSMAP)**


---

**NOTE:** On this device Group B Interrupts are not Implemented hence user software should configure only Group A interrupts.

---

**Figure 16-36. LFS Interrupt Mapping Register (LFSMAP) [offset = BCh]**


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

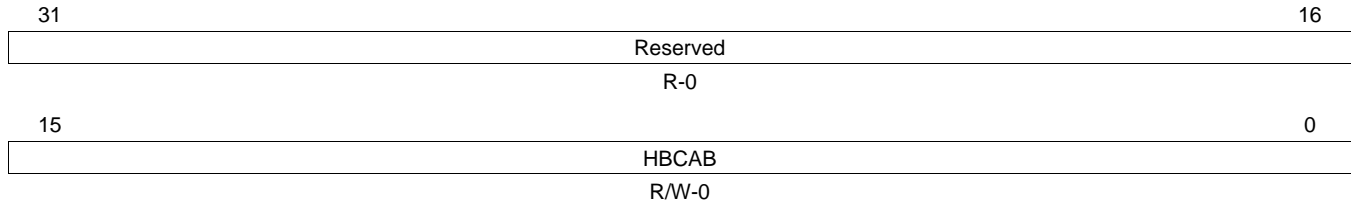
**Table 16-27. LFS Interrupt Mapping Register (LFSMAP) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	LFSAB	0	Last frame started (LFS) interrupt to Group A or Group B. The LFS interrupt of the corresponding channel is routed to Group A.
		1	The LFS interrupt of the corresponding channel is routed to Group B.

### 16.3.1.20 HBC Interrupt Mapping Register (HBCMAP)

**NOTE:** On this device Group B Interrupts are not Implemented hence user software should configure only Group A interrupts.

**Figure 16-37. HBC Interrupt Mapping Register (HBCMAP) [offset = C4h]**



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

**Table 16-28. HBC Interrupt Mapping Register (HBCMAP) Field Descriptions**

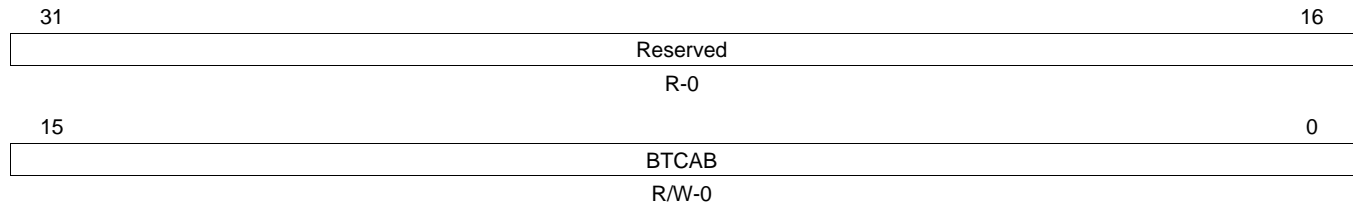
Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	HBCAB	0	Half block complete (HBC) interrupt to Group A or Group B. The HBC interrupt of the corresponding channel is routed to Group A.
		1	The HBC interrupt of the corresponding channel is routed to Group B.

**16.3.1.21 BTC Interrupt Mapping Register (BTCMAP)**


---

**NOTE:** On this device Group B Interrupts are not Implemented hence user software should configure only Group A interrupts.

---

**Figure 16-38. BTC Interrupt Mapping Register (BTCMAP) [offset = CCh]**


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

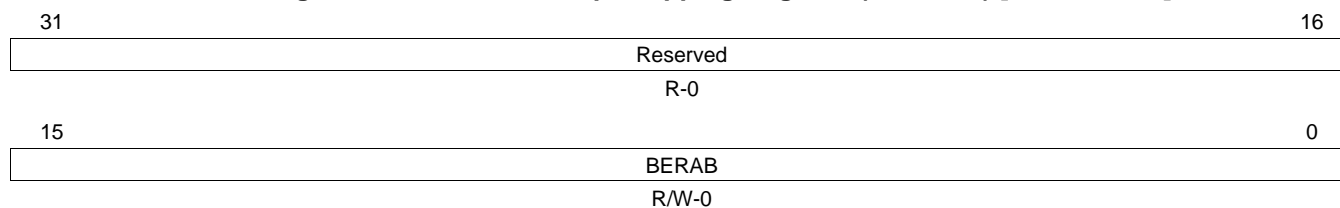
**Table 16-29. BTC Interrupt Mapping Register (BTCMAP) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	BTCAB	0	Block transfer complete (BTC) interrupt to Group A or Group B The BTC interrupt of the corresponding channel is routed to Group A.
		1	The BTC interrupt of the corresponding channel is routed to Group B.

### 16.3.1.22 BER Interrupt Mapping Register (BERMAP)

**NOTE:** On this device Group B Interrupts are not Implemented hence user software should configure only Group A interrupts.

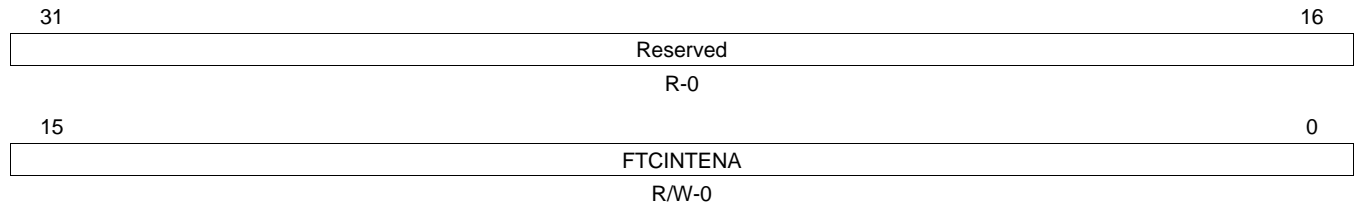
**Figure 16-39. BER Interrupt Mapping Register (BERMAP) [offset = D4h]**



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

**Table 16-30. BER Interrupt Mapping Register (BERMAP) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	BERAB	0	The BER interrupt of the corresponding channel is routed to Group A.
		1	The BER interrupt of the corresponding channel is routed to Group B.

**16.3.1.23 FTC Interrupt Enable Set (FTCINTENAS)**
**Figure 16-40. FTC Interrupt Enable Set (FTCINTENAS) [offset = DCh]**


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

**Table 16-31. FTC Interrupt Enable Set (FTCINTENAS) Field Descriptions**

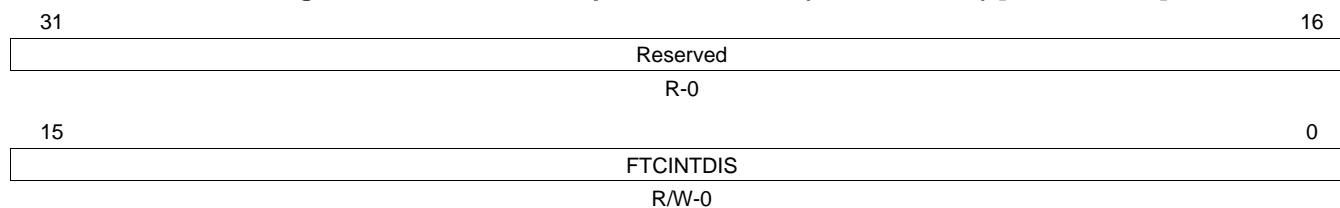
Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	FTCINTENA	0	Frame transfer complete (FTC) interrupt enable. Read: The corresponding FTC interrupt of a channel is disabled. Write: Writing a zero to this bit has no effect.
		1	Read or write: The FTC interrupt of the corresponding channel is enabled.



### 16.3.1.24 FTC Interrupt Enable Reset (FTCINTENAR)

**NOTE:** On this device Group B Interrupts are not Implemented hence user software should configure only Group A interrupts.

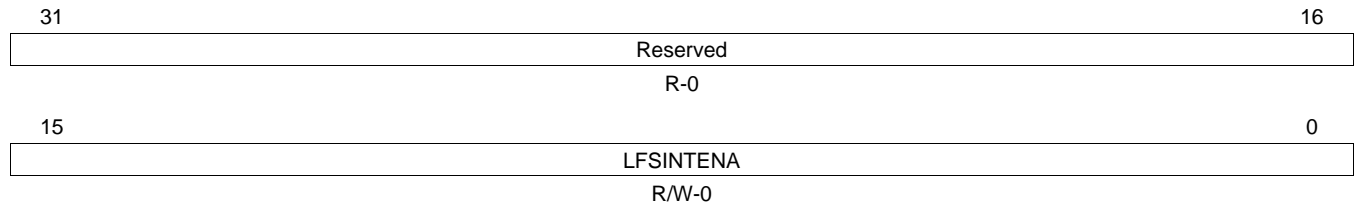
**Figure 16-41. FTC Interrupt Enable Reset (FTCINTENAR) [offset = E4h]**



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

**Table 16-32. FTC Interrupt Enable Reset (FTCINTENAR) Field Descriptions**

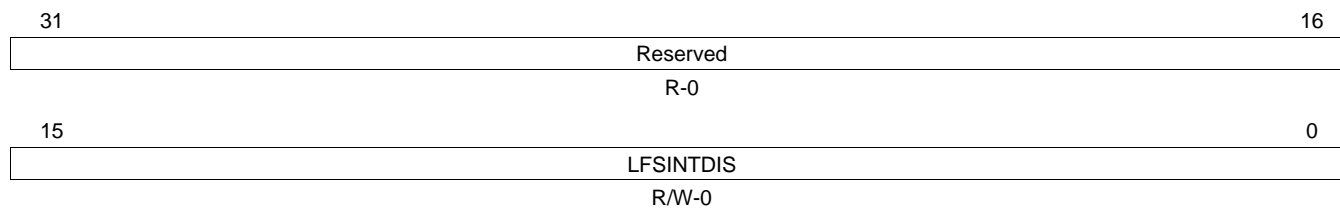
Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	FTCINTDIS	0	Frame transfer complete (FTC) interrupt disable. Read: The corresponding FTC interrupt of a channel is disabled. Write: Writing a zero to this bit has no effect.
		1	Read: The corresponding FTC interrupt of a channel is enabled. Write: The corresponding FTC interrupt is disabled.

**16.3.1.25 LFS Interrupt Enable Set (LFSINTENAS)**
**Figure 16-42. LFS Interrupt Enable Set (LFSINTENAS) [offset = ECh]**


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

**Table 16-33. LFS Interrupt Enable Set (LFSINTENAS) Field Descriptions**

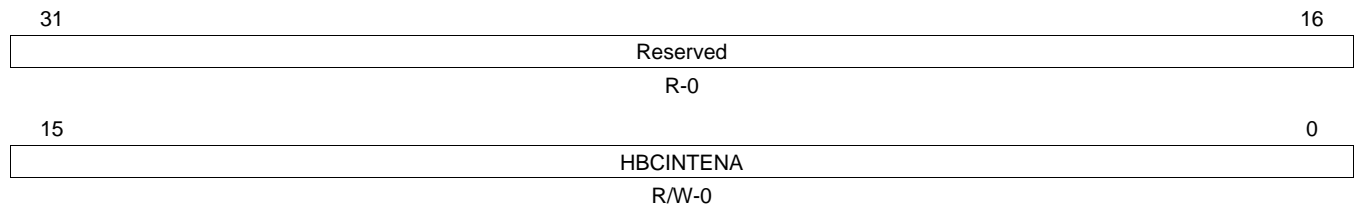
Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	LFSINTENA	0	Last frame started (LFS) interrupt enable. Read: The corresponding LFS interrupt of a channel is disabled. Write: Writing a zero to this bit has no effect.
		1	Read or write: The LFS interrupt of the corresponding channel is disabled.

**16.3.1.26 LFS Interrupt Enable Reset (LFSINTENAR)****Figure 16-43. LFS Interrupt Enable Reset (LFSINTENAR) [offset = F4h]**

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

**Table 16-34. LFS Interrupt Enable Reset (LFSINTENAR) Field Descriptions**

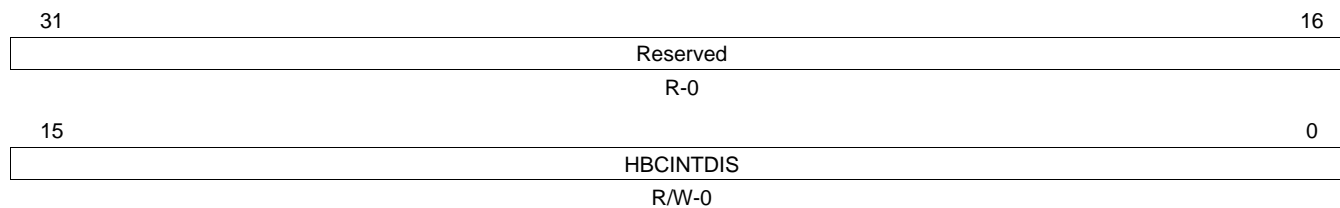
Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	LFSINTDIS	0	Last frame started (LFS) interrupt disable. Read: The LFS interrupt of the corresponding channel is disabled. Write: Writing a zero to this bit has no effect.
		1	Read: The LFS interrupt of the corresponding channel is enabled. Write: The LFS interrupt of the corresponding channel is disabled.

**16.3.1.27 HBC Interrupt Enable Reset (HBCINTENAS)**
**Figure 16-44. HBC Interrupt Enable Set (HBCINTENAS) [offset = FCh]**


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

**Table 16-35. HBC Interrupt Enable Set (HBCINTENAS) Field Descriptions**

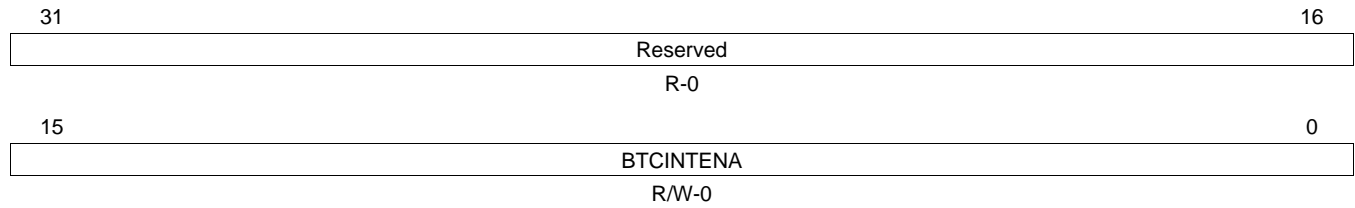
Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	HBCINTENA	0	Half block complete (HBC) interrupt enable. Read: The HBC interrupt of the corresponding channel is disabled. Write: Writing a zero to this bit has no effect.
		1	Read or write: The HBC interrupt of the corresponding channel is enabled.

**16.3.1.28 HBC Interrupt Enable Reset (HBCINTENAR)****Figure 16-45. HBC Interrupt Enable Reset (HBCINTENAR) [offset = 104h]**

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

**Table 16-36. HBC Interrupt Enable Reset (HBCINTENAR) Field Descriptions**

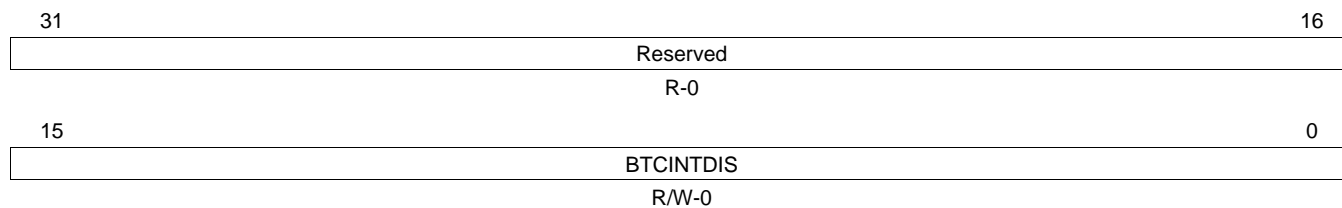
Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	HBCINTDIS	0	Half block complete (HBC) interrupt disable. Read: The HBC interrupt of the corresponding channel is disabled. Write: Writing a zero to this bit has no effect.
		1	Read: The HBC interrupt of the corresponding channel is enabled. Write: The HBC interrupt of the corresponding channel is disabled.

**16.3.1.29 BTC Interrupt Enable Set (BTCINTENAS)**
**Figure 16-46. BTC Interrupt Enable Set (BTCINTENAS) [offset = 10Ch]**


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

**Table 16-37. BTC Interrupt Enable Reset (BTCINTENAS) Field Descriptions**

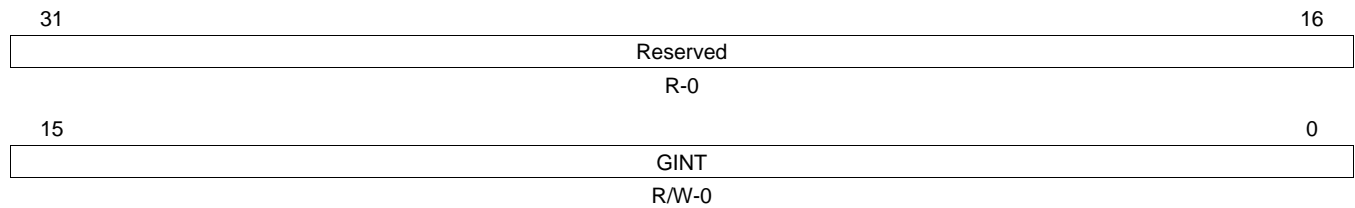
Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	BTCINTENA	0	Block transfer complete (BTC) interrupt enable. Read: The BTC interrupt of the corresponding channel is disabled. Write: Writing a zero to this bit has no effect.
		1	Read or write: The BTC interrupt of the corresponding channel is enabled.

**16.3.1.30 BTC Interrupt Enable Reset (BTCINTENAR)****Figure 16-47. BTC Interrupt Enable Reset (BTCINTENAR) [offset = 114h]**

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

**Table 16-38. BTC Interrupt Enable Reset (BTCINTENAR) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	BTCINTDIS	0	Block transfer complete (BTC) interrupt disable. Read: The BTC interrupt of the corresponding channel is enabled. Write: Writing a zero to this bit has no effect.
		1	Read: The BTC interrupt of the corresponding channel is disabled. Write: The BTC interrupt of the corresponding channel is disabled.

**16.3.1.31 Global Interrupt Flag Register (GINTFLAG)**
**Figure 16-48. Global Interrupt Flag Register (GINTFLAG) [offset = 11Ch]**


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

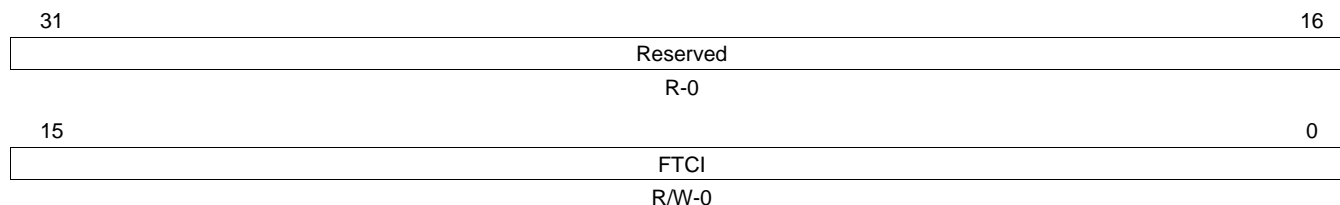
**Table 16-39. Global Interrupt Flag Register (GINTFLAG) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	GINT	0	Global interrupt flags. A global interrupt flag bit is an OR function of FTC, LFS, HBC, BTC and BER interrupt flags. No interrupt is pending on the corresponding channel.
		1	One or more of the five interrupt types (FTC, LFS, HBC, BTC, or BER) is pending on the corresponding channel.



### 16.3.1.32 FTC Interrupt Flag Register (FTCFLAG)

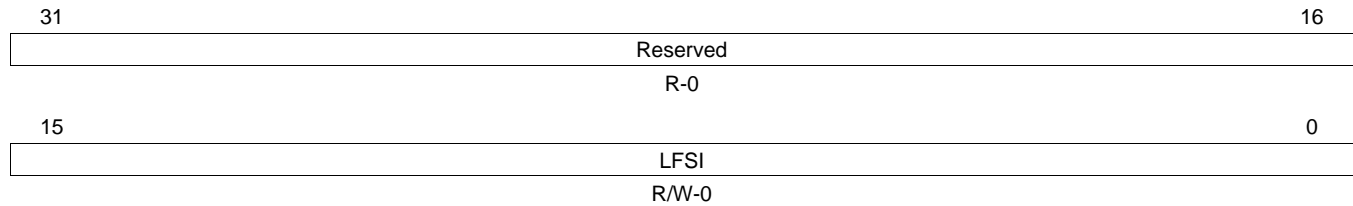
**Figure 16-49. FTC Interrupt Flag Register (FTCFLAG) [offset = 124h]**



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

**Table 16-40. FTC INTERRUPT FLAG Register (FTCFLAG) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	FTCI		<p>Frame transfer complete (FTC) flags.</p> <p><b>Note: Reading from the respective interrupt channel offset register also clears the corresponding flag (see <a href="#">Section 16.3.1.37</a> and <a href="#">Section 16.3.1.42</a>).</b></p> <p><b>Note: The state of the flag bit can be polled even if the corresponding interrupt enable bit is cleared.</b></p>
		0	<p>Read: An FTC interrupt of the corresponding channel is not pending.</p> <p>Write: Writing a zero to this bit has no effect.</p>
		1	<p>Read: An FTC interrupt of the corresponding channel is pending.</p> <p>Write: The flag is cleared.</p>

**16.3.1.33 LFS Interrupt Flag Register (LFSFLAG)**
**Figure 16-50. LFS Interrupt Flag Register (LFSFLAG) [offset = 12Ch]**


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

**Table 16-41. LFS Interrupt Flag Register (LFSFLAG) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	LFSI	0	Last frame started (LFS) flags. <b>Note: Reading from the respective interrupt channel offset register also clears the corresponding flag (see Section 16.3.1.38 and Section 16.3.1.43).</b> <b>Note: The state of the flag bit can be polled even if the corresponding interrupt enable bit is cleared.</b> Read: An LFS interrupt of the corresponding channel is not pending. Write: Writing a zero to this bit has no effect.
		1	Read: An LFS interrupt of the corresponding channel is pending. Write: The flag is cleared.

### 16.3.1.34 HBC Interrupt Flag Register (HBCFLAG)

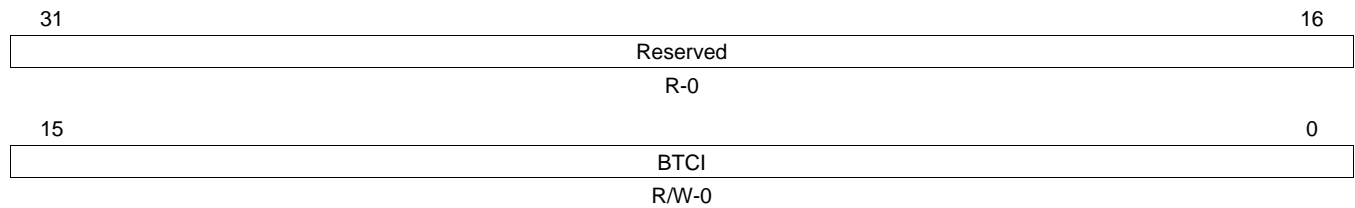
**Figure 16-51. HBC Interrupt Flag Register (HBCFLAG) [offset = 134h]**

31	Reserved	16
	R-0	
15	HBCI	0
	R/W-0	

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

**Table 16-42. HBC Interrupt Flag (HBCFLAG) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	HBCI		Half block transfer (HBC) complete flags. <b>Note: Reading from the respective interrupt channel offset register also clears the corresponding flag (see Section 16.3.1.39 and Section 16.3.1.44).</b> <b>Note: The state of the flag bit can be polled even if the corresponding interrupt enable bit is cleared.</b>
		0	Read: An HBC interrupt of the corresponding channel is not pending. Write: Writing a zero to this bit has no effect.
		1	Read: An HBC interrupt of the corresponding channel is pending. Write: The flag is cleared.

**16.3.1.35 BTC Interrupt Flag Register (BTCFLAG)**
**Figure 16-52. BTC Interrupt Flag Register (BTCFLAG) [offset = 13Ch]**


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

**Table 16-43. BTC Interrupt Flag Register (BTCFLAG) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	BTCI	0	Block transfer complete (BTC) flags. <b>Note: Reading from the respective interrupt channel offset register also clears the corresponding flag (see <a href="#">Section 16.3.1.40</a> and <a href="#">Section 16.3.1.45</a>).</b> <b>Note: The state of the flag bit can be polled even if the corresponding interrupt enable bit is cleared.</b> Read: A BTC interrupt of the corresponding channel is not pending. Write: Writing a zero to this bit has no effect.
		1	Read: A BTC interrupt of the corresponding channel is pending. Write: The flag is cleared.

### 16.3.1.36 BER Interrupt Flag Register (BERFLAG)

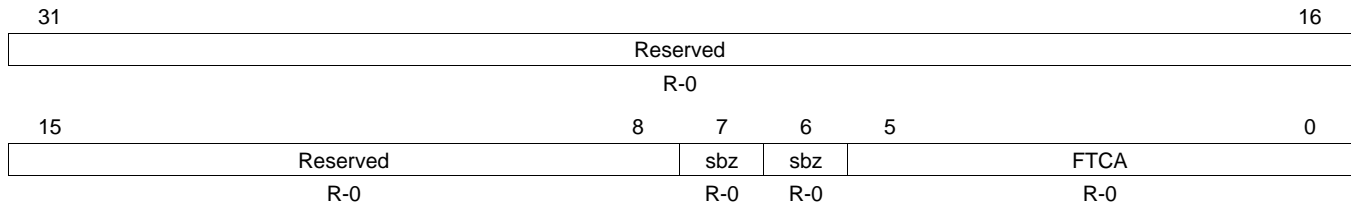
**Figure 16-53. BER Interrupt Flag Register (BERFLAG) [offset = 144h]**

31	Reserved	16
	R-0	
15	BERI	0
	R/W-0	

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

**Table 16-44. BER Interrupt Flag Register (BERFLAG) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	BERI		Bus error (BER) flags. <b>Note: Reading from the respective interrupt channel offset register also clears the corresponding flag (see Section 16.3.1.41 and Section 16.3.1.46).</b> <b>Note: The state of the flag bit can be polled even if the corresponding interrupt enable bit is cleared.</b>
		0	Read: An BER interrupt of the corresponding channel is not pending. Write: Writing a zero to this bit has no effect.
		1	Read: An BER interrupt of the corresponding channel is pending. Write: The flag is cleared.

**16.3.1.37 FTCA Interrupt Channel Offset Register (FTCAOFFSET)**
**Figure 16-54. FTCA Interrupt Channel Offset Register (FTCAOFFSET) [offset = 14Ch]**


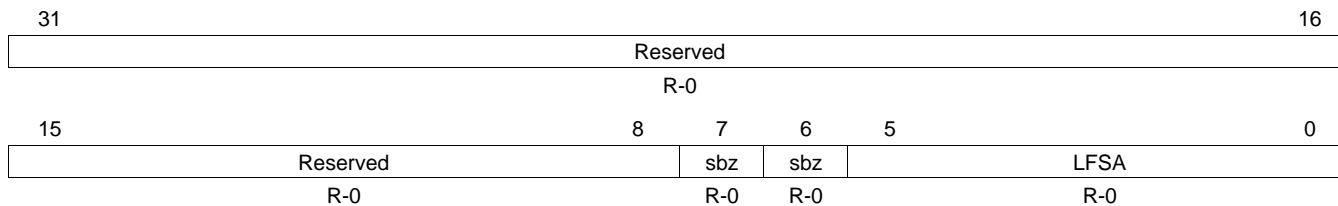
LEGEND: R = Read only; -n = value after reset

**Table 16-45. FTCA Interrupt Channel Offset Register (FTCAOFFSET) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
7-6	sbz		These bits should always be programmed as zero.
5-0	FTCA		Channel causing FTC interrupt Group A. These bits contain the channel number of the pending interrupt for Group A if the corresponding interrupt enable is set. <b>Note: Reading this location clears the corresponding interrupt pending flag (see Section 16.3.1.32) with the highest priority.</b>
		0	No interrupt is pending.
		1	Channel 0 is causing the pending interrupt Group A.
		10h	Channel 15 is causing the pending interrupt Group A.

### 16.3.1.38 LFSA Interrupt Channel Offset Register (LFSAOFFSET)

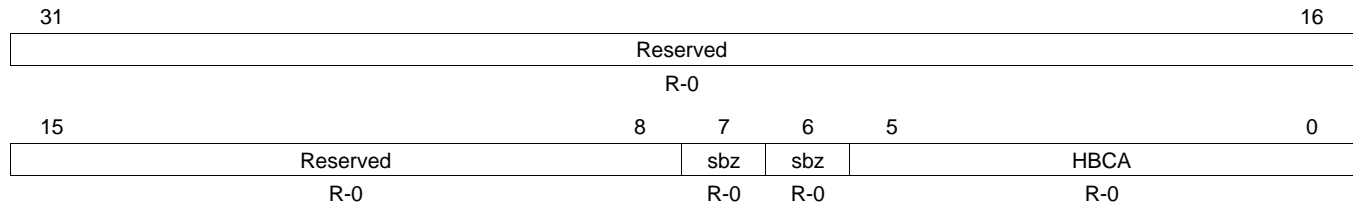
**Figure 16-55. LFSA Interrupt Channel Offset Register (LFSAOFFSET) [offset = 150h]**



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

**Table 16-46. LFSA Interrupt Channel Offset Register (LFSAOFFSET) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
7-6	sbz		These bits should always be programmed as zero.
5-0	LFSA		Channel causing LFS interrupt Group A. These bits contain the channel number of the pending interrupt for Group A if the corresponding interrupt enable is set.  <b>Note: Reading this location clears the corresponding interrupt pending flag (see <a href="#">Section 16.3.1.33</a>) with the highest priority.</b>
		0	No interrupt is pending.
		1	Channel 0 is causing the pending interrupt Group A.
		10h	Channel 15 is causing the pending interrupt Group A.

**16.3.1.39 HBCA Interrupt Channel Offset Register (HBCAOFFSET)**
**Figure 16-56. HBCA Interrupt Channel Offset Register (HBCAOFFSET) [offset = 154h]**


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

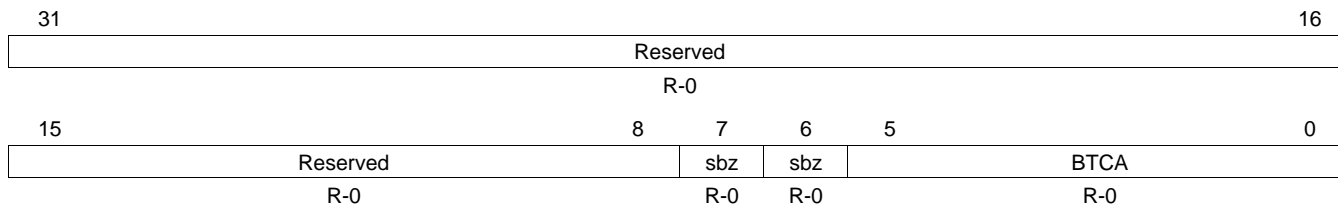
**Table 16-47. HBCA Interrupt Channel Offset Register (HBCAOFFSET) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
7-6	sbz		These bits should always be programmed as zero.
5-0	HBCA		Channel causing HBC interrupt Group A. These bits contain the channel number of the pending interrupt for Group A if the corresponding interrupt enable is set. <b>Note: Reading this location clears the corresponding interrupt pending flag (see <a href="#">Section 16.3.1.34</a>) with the highest priority.</b>
		0	No interrupt is pending.
		1	Channel 0 is causing the pending interrupt Group A.
		10h	Channel 15 is causing the pending interrupt Group A.



### 16.3.1.40 BTCA Interrupt Channel Offset Register (BTCAOFFSET)

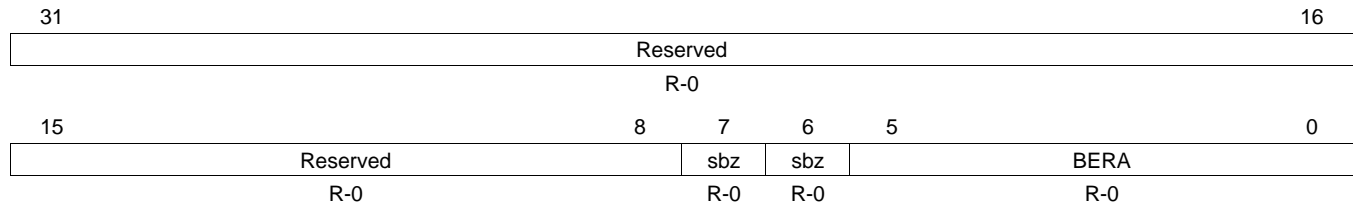
Figure 16-57. BTCA Interrupt Channel Offset Register (BTCAOFFSET) [offset = 158h]



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

Table 16-48. BTCA Interrupt Channel Offset Register (BTCAOFFSET) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
7-6	sbz		These bits should always be programmed as zero.
5-0	BTCA		Channel causing BTC interrupt Group A. These bits contain the channel number of the pending interrupt for Group A if the corresponding interrupt enable is set.  <b>Note: Reading this location clears the corresponding interrupt pending flag (see <a href="#">Section 16.3.1.35</a>) with the highest priority.</b>
		0	No interrupt is pending.
		1	Channel 0 is causing the pending interrupt Group A.
		10h	Channel 15 is causing the pending interrupt Group A.

**16.3.1.41 BERA Interrupt Channel Offset Register (BERAOFFSET)**
**Figure 16-58. BERA Interrupt Channel Offset Register (BERAOFFSET) [offset = 15Ch]**


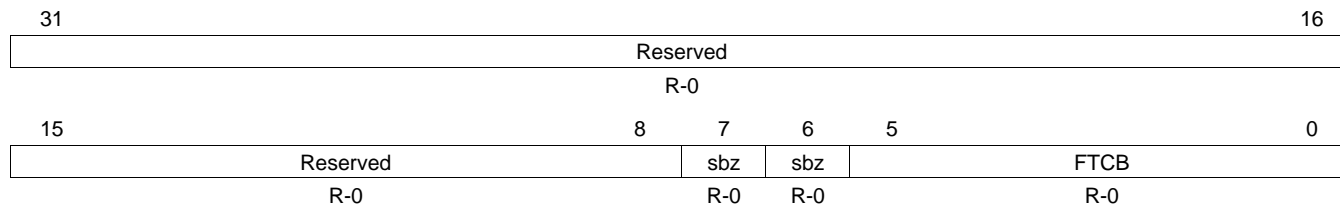
LEGEND: R = Read only; -n = value after reset

**Table 16-49. BERA Interrupt Channel Offset Register (BERAOFFSET) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
7-6	sbz		These bits should always be programmed as zero.
5-0	BERA		Channel causing BER interrupt Group A. These bits contain the channel number of the pending interrupt for Group A if the corresponding interrupt enable is set.  <b>Note: Reading this location clears the corresponding interrupt pending flag (see <a href="#">Section 16.3.1.36</a>) with the highest priority.</b>
		0	No interrupt is pending.
		1	Channel 0 is causing the pending interrupt Group A.
		10h	Channel 15 is causing the pending interrupt Group A.

### 16.3.1.42 FTCB Interrupt Channel Offset Register (FTCBOFFSET)

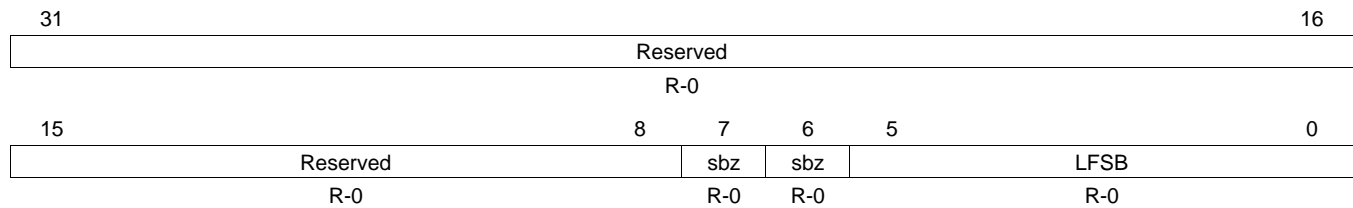
**Figure 16-59. FTCB Interrupt Channel Offset Register (FTCBOFFSET) [offset = 160h]**



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

**Table 16-50. FTCB Interrupt Channel Offset Register (FTCBOFFSET) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
7-6	sbz		These bits should always be programmed as zero.
5-0	FTCB		Channel causing FTC interrupt Group B. These bits contain the channel number of the pending interrupt for Group B if the corresponding interrupt enable is set. <b>Note: Reading this location clears the corresponding interrupt pending flag (see <a href="#">Section 16.3.1.32</a>) with the highest priority.</b>
		0	No interrupt is pending.
		1	Channel 0 is causing the pending interrupt Group B.
		10h	Channel 15 is causing the pending interrupt Group B.

**16.3.1.43 LFSB Interrupt Channel Offset Register (LFSBOFFSET)**
**Figure 16-60. LFSB Interrupt Channel Offset Register (LFSBOFFSET) [offset = 164h]**


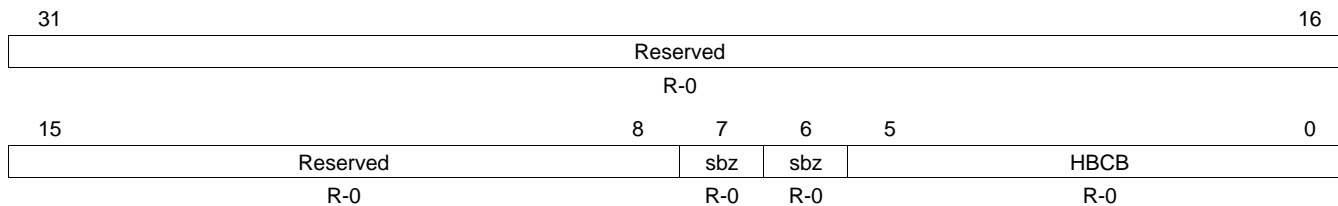
LEGEND: R = Read only; -n = value after reset

**Table 16-51. LFSB Interrupt Channel Offset Register (LFSBOFFSET) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
7-6	sbz		These bits should always be programmed as zero.
5-0	LFSB		Channel causing LFS interrupt Group B. These bits contain the channel number of the pending interrupt for Group B if the corresponding interrupt enable is set.  <b>Note: Reading this location clears the corresponding interrupt pending flag (see <a href="#">Section 16.3.1.33</a>) with the highest priority.</b>
		0	No interrupt is pending.
		1	Channel 0 is causing the pending interrupt Group B.
		10h	Channel 15 is causing the pending interrupt Group B.

### 16.3.1.44 HBCB Interrupt Channel Offset Register (HBCBOFFSET)

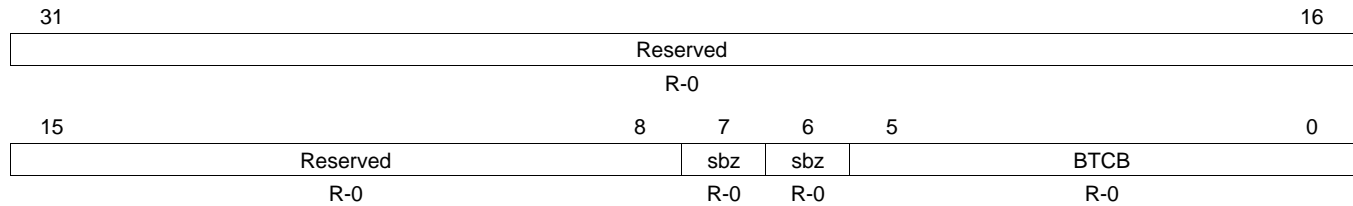
**Figure 16-61. HBCB Interrupt Channel Offset Register (HBCBOFFSET) [offset = 168h]**



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

**Table 16-52. HBCB Interrupt Channel Offset Register (HBCBOFFSET) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
7-6	sbz		These bits should always be programmed as zero.
5-0	HBCB		Channel causing HBC interrupt Group B. These bits contain the channel number of the pending interrupt for Group B if the corresponding interrupt enable is set. <b>Note: Reading this location clears the corresponding interrupt pending flag (see <a href="#">Section 16.3.1.34</a>) with the highest priority.</b>
		0	No interrupt is pending.
		1	Channel 0 is causing the pending interrupt Group B.
		10h	Channel 15 is causing the pending interrupt Group B.

**16.3.1.45 BTCB Interrupt Channel Offset Register (BTCBOFFSET)**
**Figure 16-62. BTCB Interrupt Channel Offset Register (BTCBOFFSET) [offset = 16Ch]**


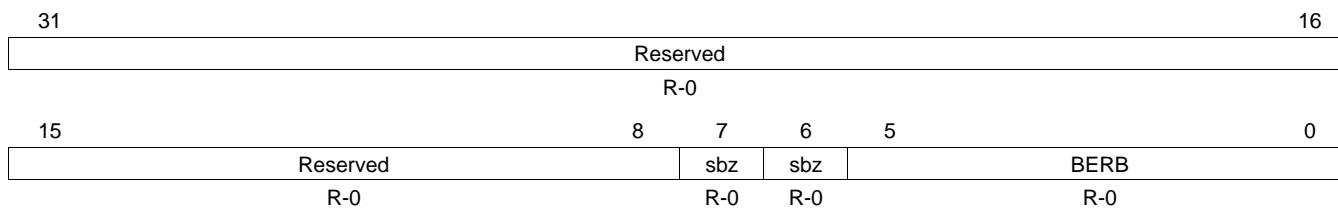
LEGEND: R = Read only; -n = value after reset

**Table 16-53. BTCB Interrupt Channel Offset Register (BTCBOFFSET) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
7-6	sbz		These bits should always be programmed as zero.
5-0	BTCB		Channel causing BTC interrupt Group B. These bits contain the channel number of the pending interrupt for Group B if the corresponding interrupt enable is set.  <b>Note: Reading this location clears the corresponding interrupt pending flag (see <a href="#">Section 16.3.1.35</a>) with the highest priority.</b>
		0	No interrupt is pending.
		1	Channel 0 is causing the pending interrupt Group B.
		10h	Channel 15 is causing the pending interrupt Group B.

### 16.3.1.46 BERB Interrupt Channel Offset Register (BERBOFFSET)

**Figure 16-63. BERB Interrupt Channel Offset Register (BERBOFFSET) [offset = 170h]**



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

**Table 16-54. BERB Interrupt Channel Offset Register (BERBOFFSET) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
7-6	sbz		These bits should always be programmed as zero.
5-0	BERB		Channel causing BER interrupt Group B. These bits contain the channel number of the pending interrupt for Group B if the corresponding interrupt enable is set.  <b>Note: Reading this location clears the corresponding interrupt pending flag (see <a href="#">Section 16.3.1.36</a>) with the highest priority.</b>
		0	No interrupt is pending.
		1	Channel 0 is causing the pending interrupt Group B.
		10h	Channel 15 is causing the pending interrupt Group B.

**16.3.1.47 Port Control Register (PTCRL)**
**Figure 16-64. Port Control Register (PTCRL) [offset = 178h]**

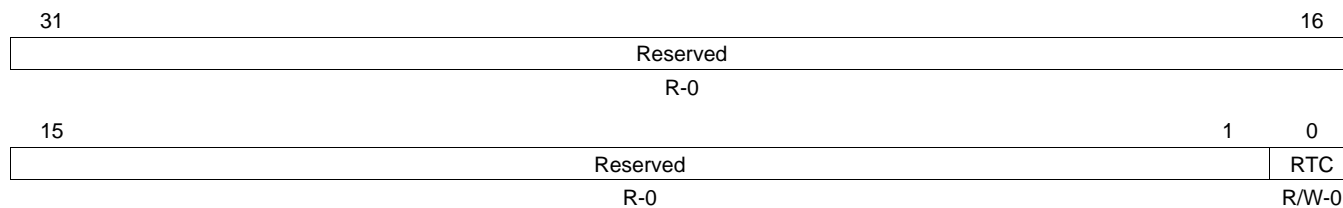
31	25	24		
Reserved		PENDB		
R-0		R-0		
23	19	18	17	16
Reserved		BYB	PSFRHQPB	PSFRLQPB
R-0		R/W-0	R/W-0	R/W-0
15				0
Reserved				
R-0				

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

**Table 16-55. Port Control Register (PTCRL) Field Descriptions**

Bit	Field	Value	Description
31-25	Reserved	0	Read returns 0. Writes have no effect.
24	PENDB	0	Transfers pending for Port B. This flag determines if transfers are ongoing on port B. The flag will be cleared if no transfers are performed. It can be used to determine if there is still data transferred while DMA_EN is set to 0 in GCTCRL. In this case, once all transfers are finished, the flag will be set to 0.
		0	No transfers are pending.
		1	Transfers are pending.
23-19	Reserved	0	Read returns 0. Writes have no effect.
18	BYB	0	Bypass FIFO B.
		0	FIFO B is not bypassed.
		1	FIFO B is bypassed. Writing 1 to this bit limits the FIFO depth to the size of one element. That means that after one element is read, the write-out to the destination will begin. This feature is particularly useful to minimize switching latency between channels.
			<b>Note: This feature does not make optimal use of bus bandwidth.</b>
17	PSFRHQPB	0	Priority scheme fix or rotate for high priority queue of Port B.
		0	Fixed priority is used.
		1	Rotation priority is used.
16	PSFRLQPB	0	Priority scheme fix or rotate for low priority queue of Port B.
		0	The fixed priority scheme is used.
		1	The rotation priority scheme is used.
15-0	Reserved	0	Read returns 0. Writes have no effect.

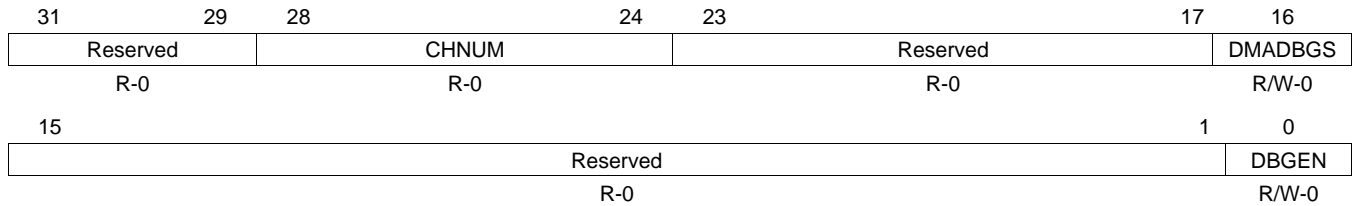


**16.3.1.48 RAM Test Control (RTCTRL)****Figure 16-65. RAM Test Control (RTCTRL) [offset = 17Ch]**

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

**Table 16-56. RAM Test Control (RTCTRL) Field Descriptions**

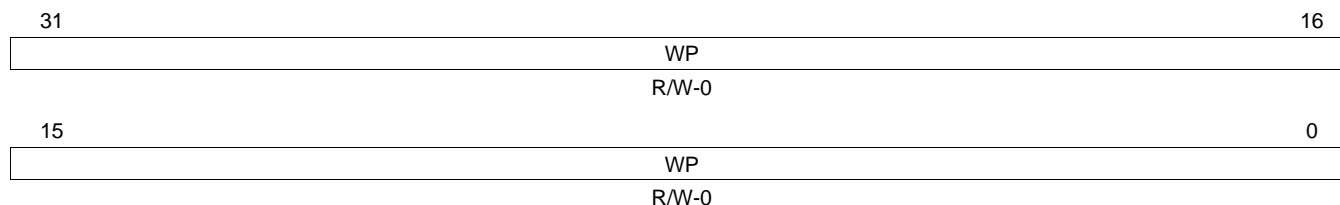
Bit	Field	Value	Description
31-1	Reserved	0	Read returns 0. Writes have no effect.
0	RTC	0	RAM Test Control Disabled.
		1	RAM Test Control Enabled..

**16.3.1.49 Debug Control (DCTRL)**
**Figure 16-66. Debug Control (DCTRL) [offset = 180h]**


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

**Table 16-57. Debug Control (DCTRL) Field Descriptions**

Bit	Field	Value	Description
31-29	Reserved	0	Read returns 0. Writes have no effect.
28-24	CHNUM	0-1Fh	Channel Number. This bit field indicates the channel number that causes the watchpoint to match.
23-17	Reserved	0	Read returns 0. Writes have no effect.
16	DMADBGS	0  1	DMA debug status. When a watchpoint is set up to watch for a unique bus address or a range of addresses is true on one of the three bus ports, then the DMA debug status bit is set to 1 and a debug request signal is asserted to the main CPU. The CPU must write a 1 to clear this bit for the DMA controller to release the debug request signal.  Read: No watchpoint condition is detected. Write: Writing a zero to this bit has no effect.  Read: The watchpoint condition is detected. Write: The bit is cleared.
15-1	Reserved	0	Read returns 0. Writes have no effect.
0	DBGEN	0  1	Debug Enable.  <b>Note: This bit can only be set when using a debugger.</b> <b>Note: This bit is reset when Test reset (nTRST) is low.</b>  Read: Debug is disabled. Write: The watchpoint checking logics is enabled.

**16.3.1.50 Watch Point Register (WPR)****Figure 16-67. Watch Point Register (WPR) [offset = 184h]**

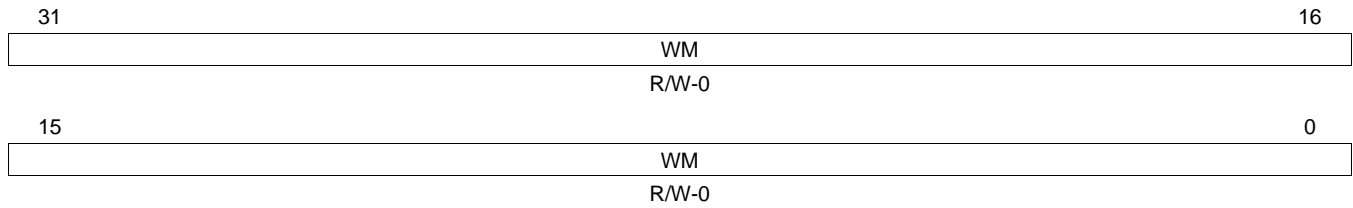
LEGEND: R = Read only; -n = value after reset

**Table 16-58. Watch Point Register (WPR) Field Descriptions**

Bit	Field	Value	Description
31-0	WP	0-FFFF FFFFh	Watch point. <b>Note: These bits can only be set when using a debugger.</b> This register is only reset by a test reset (nTRST). A 32-bit address can be programmed into this register as a watchpoint. This register is used with the watch mask register (WMR). When the DBGEN bit in the DCTRL register is set and a unique address or a range of addresses are detected on the AHB address bus of Port B, a debug request signal is sent to the ARM CPU. The state machine of the port in which the watchpoint condition is true is frozen.

**16.3.1.51 Watch Mask Register (WMR)**

**Figure 16-68. Watch Mask Register (WMR) [offset = 188h]**



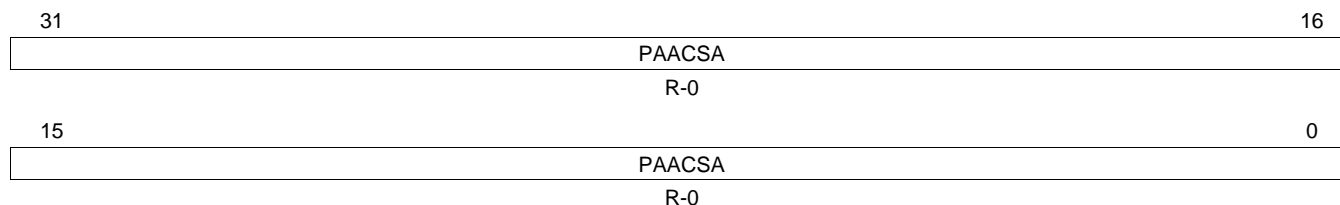
LEGEND: R = Read only; -n = value after reset

**Table 16-59. Watch Mask Register (WMR) Field Descriptions**

Bit	Field	Value	Description
31-0	WM		Watch mask. <b>Note: These bits can only be set when using a debugger.</b> This register is only reset by a test reset (nTRST).
		0	Setting a bit to 0 allows the bit in the WPR register to be used for address matching for a watch point.
		1	Setting a bit to 1 in the WMR register masks the corresponding bit in the WPR register and is disregarded in the comparison.

### 16.3.1.52 Port A Active Channel Source Address Register (PAACSADDR)

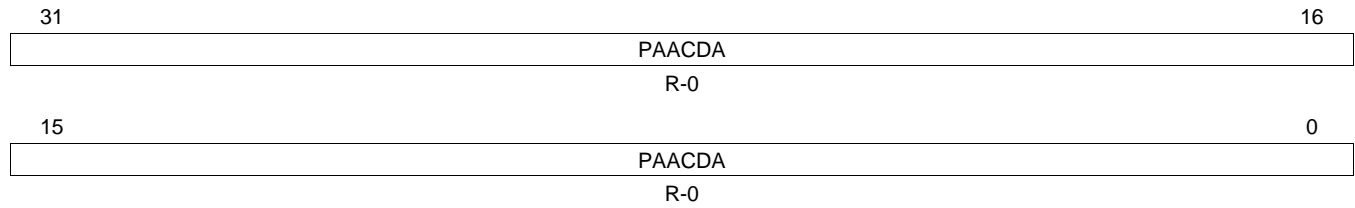
**Figure 16-69. Port A Active Channel Source Address Register (PAACSADDR) [offset = 18Ch]**



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

**Table 16-60. Port A Active Channel Source Address Register (PAACSADDR) Field Descriptions**

Bit	Field	Value	Description
31-0	PAACSA	0-FFFF FFFFh	Port A Active Channel Source Address. This register contains the current source address of the active channel as broadcasted in <a href="#">Section 16.3.1.3</a> for Port A.

**16.3.1.53 Port A Active Channel Destination Address Register (PAACDADDR)**
**Figure 16-70. Port A Active Channel Destination Address Register (PAACDADDR) [offset = 190h]**


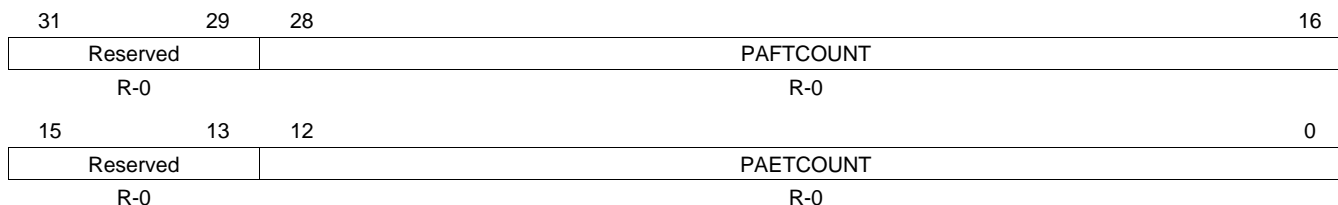
LEGEND: R = Read only; -n = value after reset

**Table 16-61. Port A Active Channel Destination Address Register (PAACDADDR) Field Descriptions**

Bit	Field	Value	Description
31-0	PAACDA	0-FFFF FFFFh	Port A Active Channel Destination Address. This register contains the current destination address of the active channel as broadcasted in <a href="#">Section 16.3.1.3</a> for Port A.

### 16.3.1.54 Port A Active Channel Transfer Count Register (PAACTC)

**Figure 16-71. Port A Active Channel Transfer Count Register (PAACTC) [offset = 194h]**



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

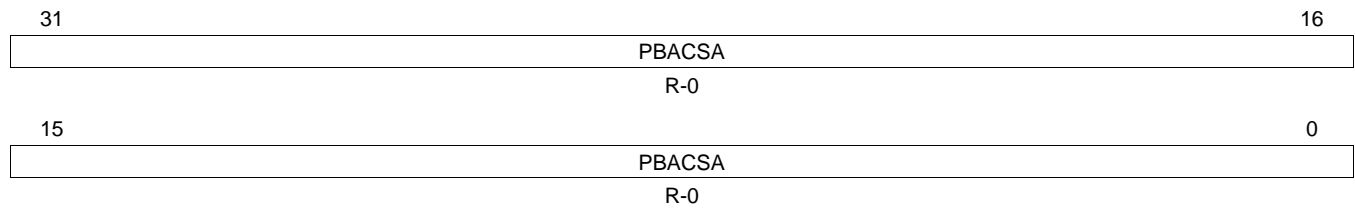
**Table 16-62. Port A Active Channel Transfer Count Register (PAACTC) Field Descriptions**

Bit	Field	Value	Description
31-29	Reserved	0	Read returns 0. Writes have no effect.
28-16	PBFTCOUNT	0-1FFFh	Port A active channel frame count. These bits contain the current frame count value of the active channel as broadcasted in <a href="#">Section 16.3.1.3</a> for Port A.
15-13	Reserved	0	Read returns 0. Writes have no effect.
12-0	PBETCOUNT	0-1FFFh	Port A active channel element count. These bits contain the current element count value of the active channel as broadcasted in <a href="#">Section 16.3.1.3</a> for Port A.

PAETCOUNT and PAFTCOUNT indicate the count values of the active channel currently residing in DMA's execution queue. With multiple pending channels and arbitration between channels, the PAETCOUNT and PAFTCOUNT should not be polled to determine the end of a channel transfer.

**16.3.1.55 Port B Active Channel Source Address Register (PBACSADDR)**

**Figure 16-72. Port B Active Channel Source Address Register (PBACDADDR) [offset = 198h]**



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

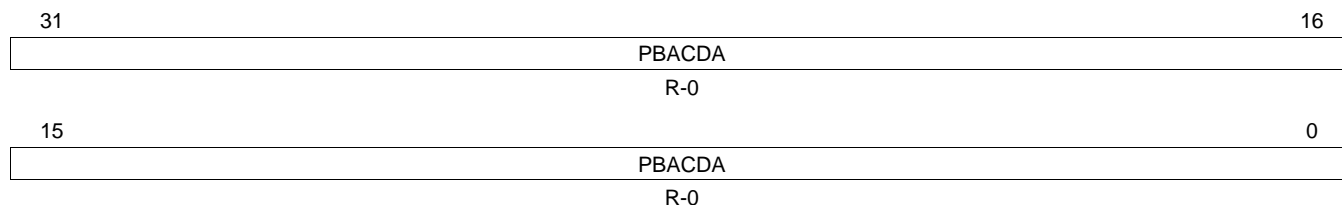
**Table 16-63. Port B Active Channel Source Address Register (PBACDADDR) Field Descriptions**

Bit	Field	Value	Description
31-0	PBACSA	0-FFFF FFFFh	Port B Active Channel Source Address. This register contains the current source address of the active channel as broadcasted in <a href="#">Section 16.3.1.3</a> for Port B.



### 16.3.1.56 Port B Active Channel Destination Address Register (PBACDADDR)

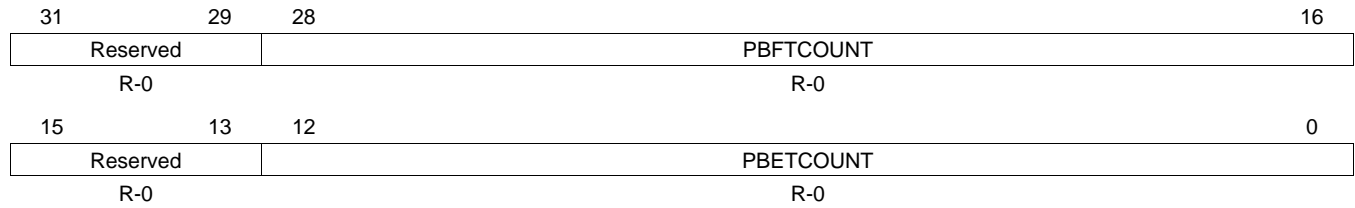
**Figure 16-73. Port B Active Channel Destination Address Register (PBACDADDR) [offset = 19Ch]**



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

**Table 16-64. Port B Active Channel Destination Address Register (PBACDADDR) Field Descriptions**

Bit	Field	Value	Description
31-0	PBACDA	0-FFFF FFFFh	Port B Active Channel Destination Address. This register contains the current destination address of the active channel as broadcasted in <a href="#">Section 16.3.1.3</a> for Port B.

**16.3.1.57 Port B Active Channel Transfer Count Register (PBACTC)**
**Figure 16-74. Port B Active Channel Transfer Count Register (PBACTC) [offset = 1A0h]**


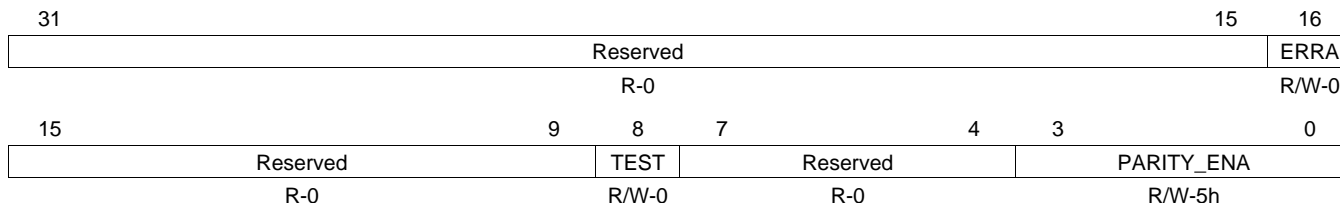
LEGEND: R = Read only; -n = value after reset

**Table 16-65. Port B Active Channel Transfer Count Register (PBACTC) Field Descriptions**

Bit	Field	Value	Description
31-29	Reserved	0	Read returns 0. Writes have no effect.
28-16	PBFTCOUNT	0-1FFFh	Port B active channel frame count. These bits contain the current frame count value of the active channel as broadcasted in <a href="#">Section 16.3.1.3</a> for Port B.
15-13	Reserved	0	Read returns 0. Writes have no effect.
12-0	PBETCOUNT	0-1FFFh	Port B active channel element count. These bits contain the current element count value of the active channel as broadcasted in <a href="#">Section 16.3.1.3</a> for Port B.

### 16.3.1.58 Parity Control Register (DMAPCR)

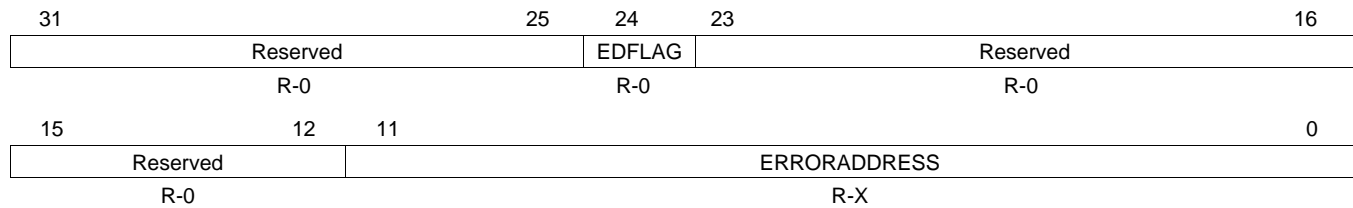
**Figure 16-75. Parity Control Register (DMAPCR) [offset = 1A8h]**



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

**Table 16-66. Parity Control Register (DMAPCR) Field Descriptions**

Bit	Field	Value	Description
31-17	Reserved	0	Read returns 0. Writes have no effect.
16	ERRA	0	Error action. If a parity error is detected on control packet x (x = 0, 1, ... n), then the enable/disable state of control packet x remains unchanged.
		1	If a parity error is detected on control packet x (x = 0, 1, ...n), then the DMA controller is disabled immediately. If a frame on control packet x is processed at the time the parity error is detected, then remaining elements of this frame will not be transferred anymore. The DMA will be disabled regardless of whether the error was detected during a read to the control packet RAM performed by the DMA state machine or by a different master.
15-9	Reserved	0	Read returns 0. Writes have no effect.
8	TEST	0	When this bit is set, the parity bits are memory mapped to make them accessible by the CPU. The parity bits are not memory mapped.
		1	The parity bits are memory mapped.
7-4	Reserved	0	Read returns 0. Writes have no effect.
3-0	PARITY_ENA	5h	Parity error detection enable. This bit field enables or disables the parity check on read operations and the parity calculation on write operations. If parity checking is enabled and a parity error is detected the DMA_UERR signal is activated. The parity check is disabled.
		All other values	The parity check is enabled. <b>Note: It is recommended to write a 5h to enable parity check, to guard against soft error from flipping PARITY_ENA to a disable state.</b>

**16.3.1.59 DMA Parity Error Address Register (DMAPAR)**
**Figure 16-76. DMA Parity Error Address Register (DMAPAR) [offset = 1ACh]**


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

**Table 16-67. DMA Parity Error Address Register (DMAPAR) Field Descriptions**

Bit	Field	Value	Description
31-25	Reserved	0	Read returns 0. Writes have no effect.
24	EDFLAG	0	Parity Error Detection Flag. This flag indicates if a parity error occurred on reading DMA Control packet RAM. Read: No error occurred. Write: No effect.
		1	Read: Error detected and the address is captured in DMAPAR's ERROR_ADDRESS field Write: Clears the bit.
23-12	Reserved	0	Read returns 0. Writes have no effect.
11-0	ERRORADDRESS	0-FFFh	Error address. These bits hold the address of the first parity error generated in the RAM. This error address is frozen from being updated until it is read by the CPU. During emulation mode when SUSPEND is high, this address is frozen even when read. <b>Note: The error address register will not be reset by PORRST nor by any other reset source.</b>

### 16.3.1.60 DMA Memory Protection Control Register (DMAMPCTRL)

Figure 16-77.

31	29	28	27	26	25	24
Reserved		INT3AB	INT3ENA	REG3AP		REG3ENA
R-0		R/W-0	R/W-0	R/W-0		R/W-0
23	21	20	19	18	17	16
Reserved		INT2AB	INT2ENA	REG2AP		REG2ENA
R-0		R/W-0	R/W-0	R/W-00		R/W-0
15	13	12	11	10	9	8
Reserved		INT1AB	INT1ENA	REG1AP		REG1ENA
R-0		R/W-0	R/W-0	R/W-0		R/W-0
7	5	4	3	2	1	0
Reserved		INT0AB	INT0ENA	REG0AP		REG0ENA
R-0		R/W-0	R/W-0	R/W-00		R/W-0

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

Table 16-68. DMA Memory Protection Control Register

Bit	Field	Value	Description
31-29	Reserved	0	Read returns 0. Writes have no effect.
28	INT3AB	0	Interrupt assignment of region 3 to Group A or Group B. The interrupt is routed to the VIM (Group A).
		1	The interrupt is routed to the second CPU (Group B).
27	INT3ENA	0	Interrupt enable of region 3. The interrupt is disabled.
		1	The interrupt is enabled.
26-25	REG3AP	0	Region 3 access permission. These bits determine the access permission for region 3 All accesses are allowed.
		1h	Read only accesses are allowed.
		2h	Write only accesses are allowed.
		3h	No accesses are allowed.
24	REG3ENA	0	Region 3 enable. The region is disabled (no address checking done).
		1	The region is enabled (address and access permission checking done).
23-21	Reserved	0	Read returns 0. Writes have no effect.
20	INT2AB	1	Interrupt assignment of region 2 to Group A or Group B. The interrupt is routed to the VIM (Group A).
		0	The interrupt is routed to the second CPU (Group B).
19	INT2ENA	1	Interrupt enable of region 2. The interrupt is disabled.
		0	The interrupt is enabled.
18-17	REG2AP	0	Region 2 access permission. These bits determine the access permission for region 2. All accesses are allowed.
		1h	Read only accesses are allowed.
		2h	Write only accesses are allowed.
		3h	No accesses are allowed.
16	REG2ENA	0	Region 2 enable. The region is disabled (no address checking done).
		1	The region is enabled (address and access permission checking done).

**Table 16-68. DMA Memory Protection Control Register (continued)**

Bit	Field	Value	Description
15-13	Reserved	0	Reads return 0. Writes have no effect.
12	INT1AB	0	Interrupt assignment of region 1 to Group A or Group B. The interrupt is routed to the VIM (Group A).
		1	The interrupt is routed to the second CPU (Group B).
11	INT1ENA	0	Interrupt enable of region 1. The interrupt is disabled.
		1	The interrupt is enabled.
10-9	REG1AP	0	Region 1 access permission. These bits determine the access permission for region 3 All accesses are allowed.
		1h	Read only accesses are allowed.
		2h	Write only accesses are allowed.
		3h	No accesses are allowed.
8	REG1ENA	0	Region 1 enable. The region is disabled (no address checking done).
		1	The region is enabled (address and access permission checking done).
7-5	Reserved	0	Reads return zeros and writes have no effect.
4	INT0AB	0	Interrupt assignment of region 0 to Group A or Group B. The interrupt is routed to the VIM (Group A).
		1	The interrupt is routed to the DSP CPU (Group B).
3	INT0ENA	0	Interrupt enable of region 0. The interrupt is disabled.
		1	The interrupt is enabled.
2-1	REG0AP	0	Region 0 access permission. These bits determine the access permission for region 0. All accesses are allowed.
		1h	Read only accesses are allowed.
		2h	Write only accesses are allowed.
		3h	No accesses are allowed.
0	REG0ENA	0	Region 0 enable. The region is disabled (no address checking done).
		1	The region is enabled (address and access permission checking done).

### 16.3.1.61 DMA Memory Protection Status Register (DMAMPST)

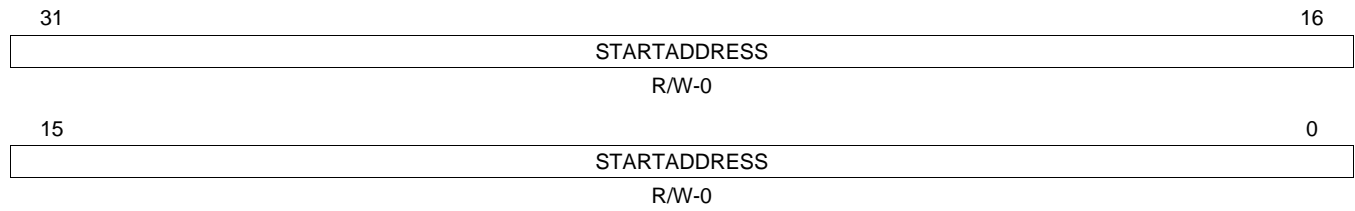
**Figure 16-78. DMA Memory Protection Status Register (DMAMPST) [offset = 1B4h]**

31	25	24	23	17	16
Reserved		REG3 FT	Reserved		REG2 FT
R-0		R/C-0	R-0		R/C-0
15	9	8	7	1	0
Reserved		REG1 FT	Reserved		REG0 FT
R-0		R/C-0	R-0		R/C-0

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

**Table 16-69. DMA Memory Protection Status Register (DMAMPST) Field Descriptions**

Bit	Field	Value	Description
31-25	Reserved	0	Read returns 0. Writes have no effect.
24	REG3FT	0	Region 3 fault. This bit determines whether an access permission violation was detected in this region. No fault was detected.
		1	Read: A fault was detected. Write: The bit was cleared.
23-17	Reserved	0	Read returns 0. Writes have no effect.
16	REG2FT	0	Region 2 fault. This bit determines whether a access permission violation was detected in this region. No fault was detected.
		1	Read: A fault was detected. Write: The bit was cleared.
15-9	Reserved	0	Read returns 0. Writes have no effect.
8	REG1FT	0	Region 1 fault. This bit determines whether an access permission violation was detected in this region. No fault was detected.
		1	Read: A fault was detected. Write: The bit was cleared.
7-1	Reserved	0	Read returns 0. Writes have no effect.
0	REG0FT	0	Region 0 fault. This bit determines whether a access permission violation was detected in this region. No fault was detected.
		1	Read: A fault was detected. Write: The bit was cleared.

**16.3.1.62 DMA Memory Protection Region 0 Start Address Register (DMAMPR0S)**
**Figure 16-79. DMA Memory Protection Region 0 Start Address Register (DMAMPR0S) [offset = 1B8h]**


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

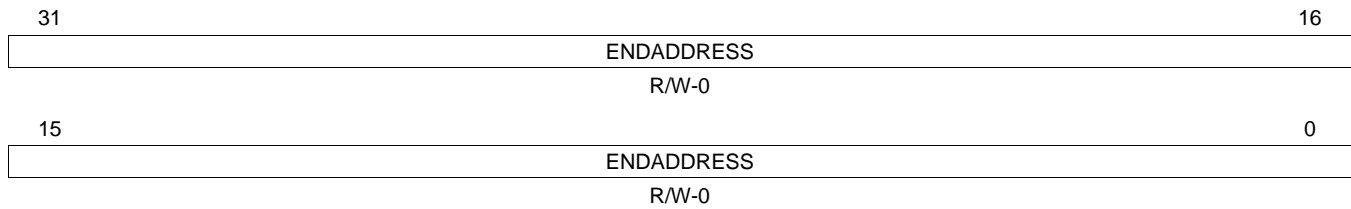
**Table 16-70. DMA Memory Protection Region 0 Start Address Register (DMAMPR0S) Field Descriptions**

Bit	Field	Value	Description
31-0	STARTADDRESS	0-FFFF FFFFh	Start Address defines the address at which the region begins.



### 16.3.1.63 DMA Memory Protection Region 0 End Address Register (DMAMPR0E)

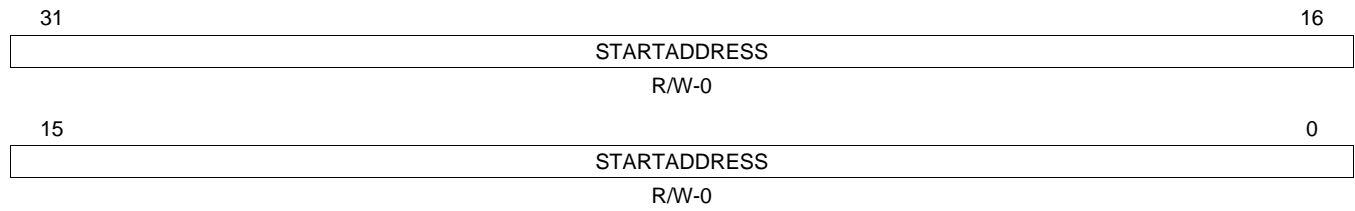
**Figure 16-80. DMA Memory Protection Region 0 End Address Register (DMAMPR0E) [offset = 1BCh]**



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

**Table 16-71. DMA Memory Protection Region 0 End Address Register (DMAMPR0E)] Field Descriptions**

Bit	Field	Value	Description
31-0	ENDADDRESS	0-FFFF FFFFh	End Address defines the address at which the region ends. The end address usually is larger than the start address for this region; otherwise, the region will wrap around at the end of the address space.

**16.3.1.64 DMA Memory Protection Region 1 Start Address Register (DMAMPR1S)**
**Figure 16-81. DMA Memory Protection Region 1 Start Address Register (DMAMPR1S) [offset = 1C0h]**


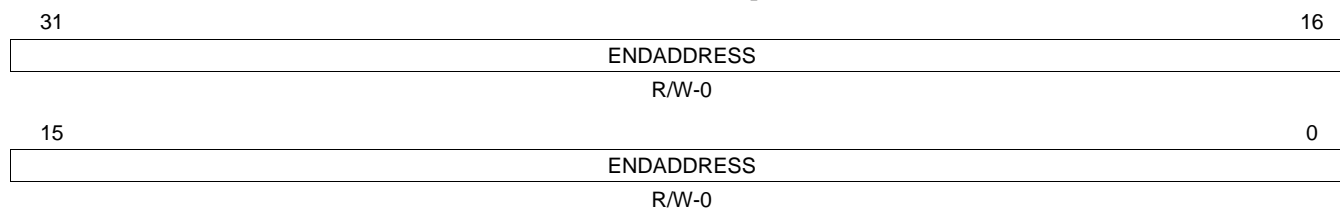
LEGEND: R = Read only; -n = value after reset

**Table 16-72. DMA Memory Protection Region 1 Start Address Register (DMAMPR1S) Field Descriptions**

Bit	Field	Value	Description
31-0	STARTADDRESS	0-FFFF FFFFh	Start Address defines the address at which the region begins.

### 16.3.1.65 DMA Memory Protection Region 1 End Address Register (DMAMPR1E)

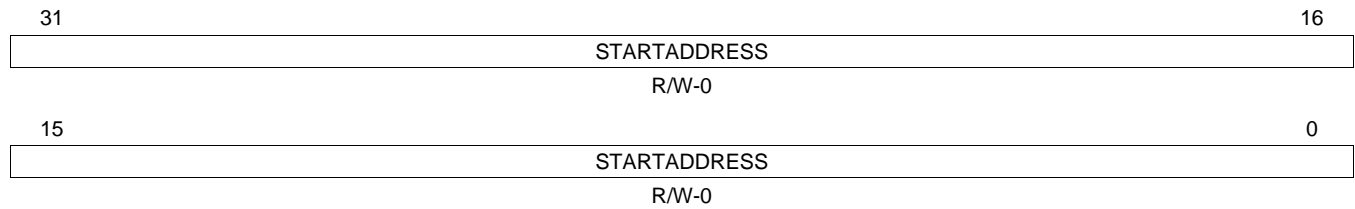
**Figure 16-82. DMA Memory Protection Region 1 End Address Register (DMAMPR1E) [offset = 1C4h]**



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

**Table 16-73. DMA Memory Protection Region 1 End Address Register (DMAMPR1E) Field Descriptions**

Bit	Field	Value	Description
31-0	ENDADDRESS	0-FFFF FFFFh	End Address defines the address at which the region ends. The end address usually is larger than the start address for this region; otherwise the region will wrap around at the end of the address space.

**16.3.1.66 DMA Memory Protection Region 2 Start Address Register (DMAMPR2S)**
**Figure 16-83. DMA Memory Protection Region 2 Start Address Register (DMAMPR2S) [offset = 1C8h]**


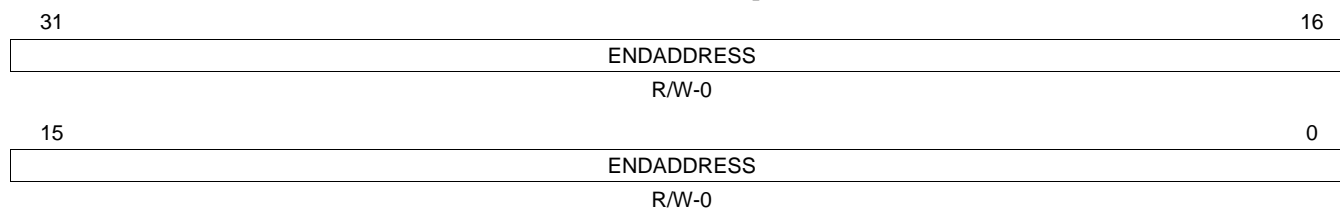
LEGEND: R = Read only; -n = value after reset

**Table 16-74. DMA Memory Protection Region 2 Start Address Register (DMAMPR2S) Field Descriptions**

Bit	Field	Value	Description
31-0	STARTADDRESS	0-FFFF FFFFh	Start Address defines the address at which the region begins.

### 16.3.1.67 DMA Memory Protection Region 2 End Address Register (DMAMPR2E)

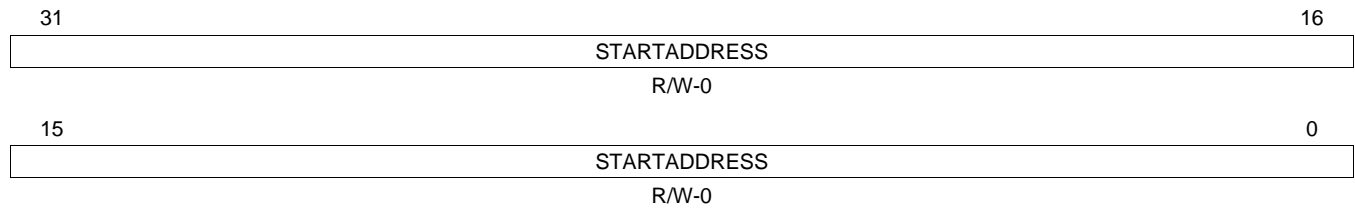
**Figure 16-84. DMA Memory Protection Region 2 End Address Register (DMAMPR2E) [offset = 1CCh]**



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

**Table 16-75. DMA Memory Protection Region 2 End Address Register (DMAMPR2E) Field Descriptions**

Bit	Field	Value	Description
31-0	ENDADDRESS	0-FFFF FFFFh	End Address defines the address at which the region ends. The end address usually is larger than the start address for this region; otherwise the region will wrap around at the end of the address space.

**16.3.1.68 DMA Memory Protection Region 3 Start Address Register (DMAMPR3S)**
**Figure 16-85. DMA Memory Protection Region 3 Start Address Register (DMAMPR3S) [offset = 1D0h]**


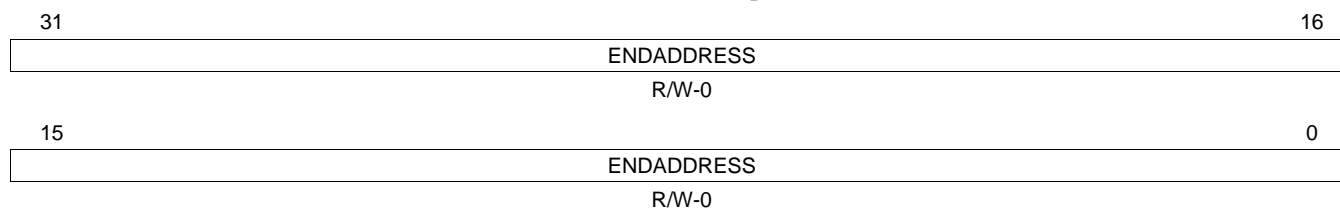
LEGEND: R = Read only; -n = value after reset

**Table 16-76. DMA Memory Protection Region 3 Start Address Register (DMAMPR3S) Field Descriptions**

Bit	Field	Value	Description
31-0	STARTADDRESS	0-FFFF FFFFh	Start Address defines the address at which the region begins.

### 16.3.1.69 DMA Memory Protection Region 3 End Address Register (DMAMPR3E)

**Figure 16-86. DMA Memory Protection Region 3 End Address Register (DMAMPR3E) [offset = 1D4h]**



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

**Table 16-77. DMA Memory Protection Region 3 End Address Register (DMAMPR3E) Field Descriptions**

Bit	Field	Value	Description
31-0	ENDADDRESS	0-FFFF FFFFh	End Address defines the address at which the region ends. The end address usually is larger than the start address for this region; otherwise, the region will wrap around at the end of the address space.

### 16.3.2 Channel Configuration

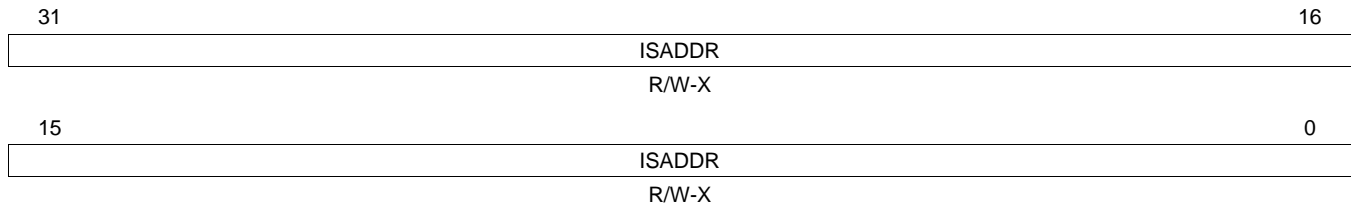
The channel configuration is defined by the channel control packet: channel control, transfer count, index pointers, source/destination address).

- It is stored in local RAM, which is protected by parity.
- Each control packet contains a total of nine fields.
- The first six fields are programmable, while the last three fields are read only.
- The RAM is accessible by queue A and queue B state machines as well as CPU.
- When there are simultaneous accesses, the priority is resolved in a fixed priority scheme with the CPU having the highest priority.

All the control packets look the same. Following, there is the detailed layout of these registers shown for control packet 0.

#### 16.3.2.1 Initial Source Address (ISADDR)

**Figure 16-87. Initial Source Address (ISADDR) [offset = 00]**



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

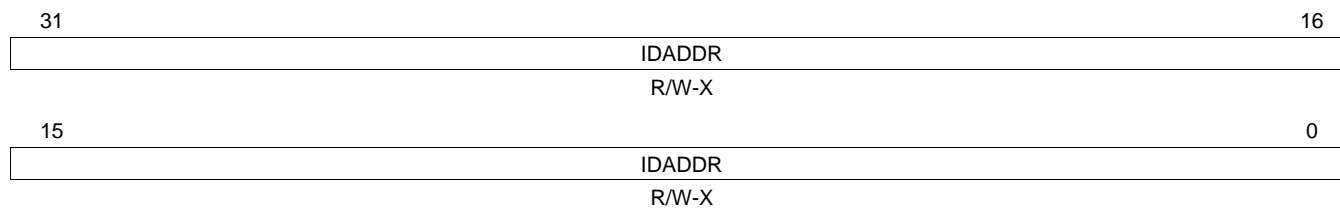
**Table 16-78. Initial Source Address (ISADDR) Field Descriptions**

Bit	Field	Value	Description
31-0	ISADDR	0-FFFF FFFFh	Initial source address. These bits give the absolute 32-bit source address (physical).



### 16.3.2.2 Initial Destination Address Register (IDADDR)

**Figure 16-88. Initial Destination Address Register (IDADDR) [offset = 04h]**



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

**Table 16-79. Initial Destination Address Register (IDADDR) Field Descriptions**

Bit	Field	Value	Description
31-0	IDADDR	0-FFFF FFFFh	Initial destination address. These bits give the absolute 32-bit destination address (physical).

**16.3.2.3 Initial Transfer Count Register (ITCOUNT)**
**Figure 16-89. Initial Transfer Count Register (ITCOUNT) [offset = 08h]**

31	29	28	16
Reserved		IFTCOUNT	
R-X		R/W-X	
15	13	12	0
Reserved		IETCOUNT	
R-X		R/W-X	

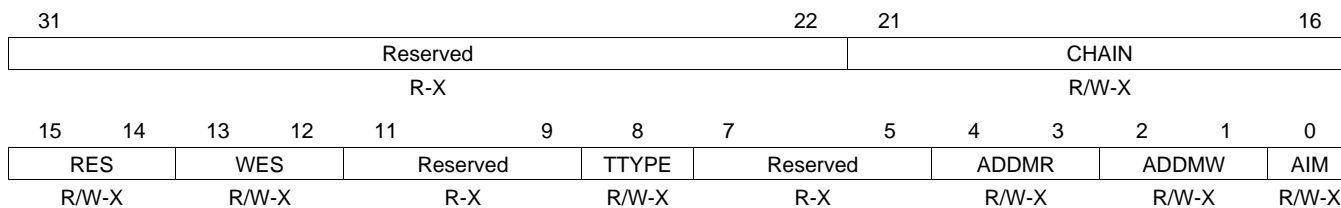
LEGEND: R = Read only; -n = value after reset; X = Unknown

**Table 16-80. Initial Transfer Count Register (ITCOUNT) Field Descriptions**

Bit	Field	Value	Description
31-29	Reserved	0	Reads are undefined. Writes have no effect.
28-16	IFTCOUNT	0-1FFFh	Initial frame transfer count. These bits define the number of frame transfers.
15-13	Reserved	0	Reads are undefined. Writes have no effect.
12-0	IETCOUNT	0-1FFFh	Initial element transfer count. These bits define the number of element transfers. The block transfer size will be IETCOUNT x IFTCOUNT

### 16.3.2.4 Channel Control Register (CHCTRL)

**Figure 16-90. Channel Control Register (CHCTRL) [offset = 10h]**



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

**Table 16-81. CHANNEL CONTROL Register (CHCTRL) Field Descriptions**

Bit	Field	Value	Description
31-22	Reserved	0	Reads are undefined. Writes have no effect.
21-16	CHAIN	0 1h 4h 4FEh	Next channel to be triggered. At the end of the programmed number of frames, the specified channel will be triggered.  <b>Note: The programmer must program the CHAIN bits before initiating a DMA transfer.</b> No channel is selected. Channel 0 is selected. Channel 15 is selected. Reserved
15-14	RES	0 1h 2h 3h	Read element size. The element is byte, 8-bit. The element is half-word, 16-bit. The element is word, 32-bit. The element is double-word, 64-bit.
13-12	WES	0 1h 2h 3h	Write element size. The element is byte, 8-bit. The element is half-word, 16-bit. The element is word, 32-bit. The element is double-word, 64-bit.
11-9	Reserved	0	Reads are undefined. Writes have no effect.
8	TTYPE	0 1	Transfer type. A hardware request triggers one frame transfer. A hardware request triggers one block transfer.
7-5	Reserved	0	Reads are undefined. Writes have no effect.
4-3	ADDMR	0 1h 2h 3h	Addressing mode read. Constant Post-increment Reserved Indexed
2-1	ADDMW	0 1h 2h 3h	Addressing mode write. Constant Post-increment Reserved Indexed
0	AIM	0 1	Auto-initiation mode. Auto-initiation mode is enabled. Single block transfer mode is enabled.

**16.3.2.5 Element Index Offset Register (EIOFF)**
**Figure 16-91. Element Index Offset Register (EIOFF) [offset = 14h]**

31	29	28	16
Reserved		EIDX	
R-X		R/W-X	
15	13	12	0
Reserved		EIDX	
R-X		R/W-X	

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

**Table 16-82. Element Index Offset Register (EIOFF) Field Descriptions**

Bit	Field	Value	Description
31-29	Reserved	0	Reads are undefined. Writes have no effect.
28-16	EIDX	0-1FFFh	Destination address element index. These bits define the offset to be added to the destination address after each element transfer.
15-13	Reserved	0	Reads are undefined. Writes have no effect.
12-0	EIDX	0-1FFFh	Source address element index. These bits define the offset to be added to the source address after each element transfer.

### 16.3.2.6 Frame Index Offset Register (FIOFF)

**Figure 16-92. Frame Index Offset Register (FIOFF) [offset = 18h]**

31	29	28	16
Reserved		FIDX	
R-X		R/W-X	
15	13	12	0
Reserved		FIDX	
R-X		R/W-X	

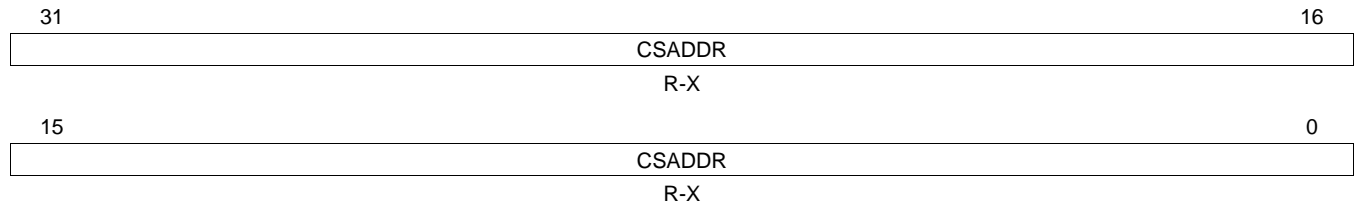
LEGEND: R = Read only; -n = value after reset; X = Unknown

**Table 16-83. Frame Index Offset Register (FIOFF) Field Descriptions**

Bit	Field	Value	Description
31-29	Reserved	0	Reads are undefined. Writes have no effect.
28-16	FIDX	0-1FFFh	Destination address frame index. These bits define the offset to be added to the destination address after element count reached one.
15-13	Reserved	0	Reads are undefined. Writes have no effect.
12-0	FIDX	0-1FFFh	Source address frame index. These bits define the offset to be added to the source address after element count reached one.

**16.3.2.7 Current Source Address Register (CSADDR)**

**Figure 16-93. Current Source Address Register (CSADDR) [offset = 800h]**



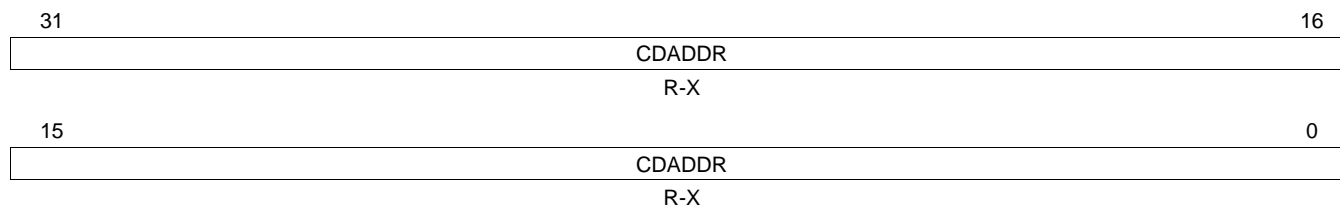
LEGEND: R = Read only; -n = value after reset; X = Unknown

**Table 16-84. Current Source Address Register (CSADDR) Field Descriptions**

Bit	Field	Value	Description
31-0	CSADDR	0-FFFF FFFFh	Current source address. These bits contain the current working absolute 32-bit source address (physical). These bits are only updated after a channel is arbitrated out from the priority queue.

### 16.3.2.8 Current Destination Address Register (CDADDR)

**Figure 16-94. Current Destination Address Register (CDADDR) [offset = 804h]**



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

**Table 16-85. Current Destination Address Register (CDADDR) Field Descriptions**

Bit	Field	Value	Description
31-0	CDADDR	0-FFFF FFFFh	Current destination address. These bits contain the current working absolute 32-bit destination address (physical). These bits are only updated after a channel is arbitrated out of the priority queue.

**16.3.2.9 Current Transfer Count Register (CTCOUNT)**
**Figure 16-95. Current Transfer Count Register (CTCOUNT) [offset = 808h]**

31	29	28	16
Reserved		CFTCOUNT	
R-X		R-X	
15	13	12	0
Reserved		CETCOUNT	
R-X		R-X	

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

**Table 16-86. Current Transfer Count Register (CTCOUNT) Field Descriptions**

Bit	Field	Value	Description
31-29	Reserved	0	Reads are undefined. Writes have no effect.
28-16	CFTCOUNT	0-1FFFh	Current frame transfer count. Returned the current remaining frame counts.
15-13	Reserved	0	Reads are undefined. Writes have no effect.
12-0	CETCOUNT	0-1FFFh	Current element transfer count. These bits return the current remaining element counts. CTCOUNT register is only updated after a channel is arbitrated out of the priority queue.





---

---

## ***External Memory Interface (EMIF)***

---

---

This chapter describes the external memory Interface (EMIF).

<b>Topic</b>	<b>Page</b>
<b>17.1 Introduction .....</b>	<b>744</b>
<b>17.2 EMIF Module Architecture .....</b>	<b>745</b>
<b>17.3 Registers .....</b>	<b>778</b>
<b>17.4 Example Configuration .....</b>	<b>792</b>

## 17.1 Introduction

### 17.1.1 Purpose of the Peripheral

This EMIF memory controller is compliant with the JESD21-C SDR SDRAM memories utilizing a 16-bit data bus. The purpose of this EMIF is to provide a means for the CPU to connect to a variety of external devices including:

- Single data rate (SDR) SDRAM
- Asynchronous devices including NOR Flash and SRAM

The most common use for the EMIF is to interface with both a flash device and an SDRAM device simultaneously. contains an example of operating the EMIF in this configuration.

### 17.1.2 Features

The EMIF includes many features to enhance the ease and flexibility of connecting to external SDR SDRAM and asynchronous devices.

#### 17.1.2.1 Asynchronous Memory Support

EMIF supports asynchronous:

- SRAM memories
- NOR Flash memories

The EMIF data bus width is up to 16 bits and there are up to 22 address lines. There is an external wait input that allows slower asynchronous memories to extend the memory access. The EMIF module supports up to 3 chip selects (EMIF\_nCS[4:2]). Each chip select has the following individually programmable attributes:

- Data Bus Width
- Read cycle timings: setup, hold, strobe
- Write cycle timings: setup, hold, strobe
- Bus turn-around time
- Extended Wait Option with Programmable Timeout
- Select Strobe option

#### 17.1.2.2 Synchronous DRAM Memory Support

The EMIF module supports 16-bit SDRAM in addition to the asynchronous memories listed in [Section 17.1.2.1](#). It has a single SDRAM chip select (EMIF\_nCS[0]). SDRAM configurations that are supported are:

- One, Two and Four Bank SDRAM devices
- Devices with Eight, Nine, Ten, and Eleven Column Address
- CAS Latency of two or three clock cycles
- 16-bit Data Bus Width
- 3.3V LVCMOS Interface

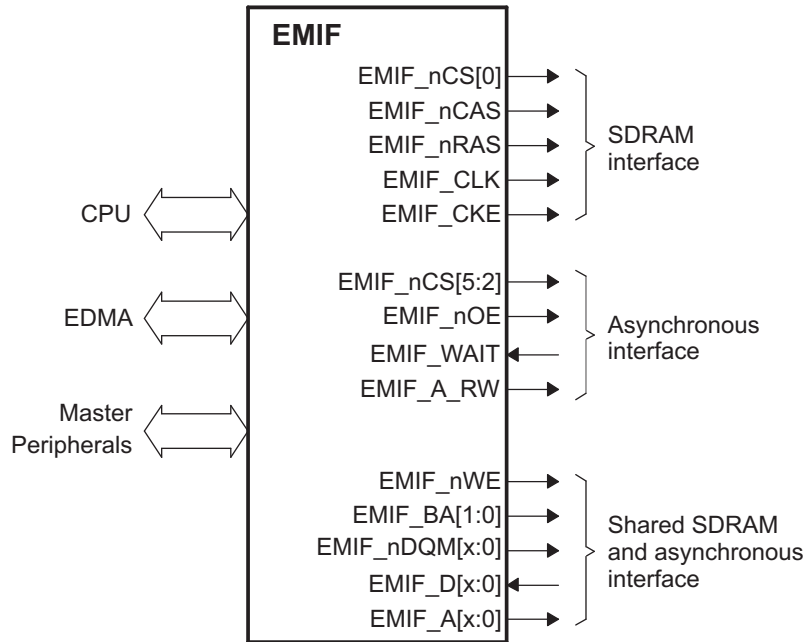
Additionally, the EMIF supports placing the SDRAM in Self-Refresh and Powerdown modes. Self-refresh mode allows the SDRAM to be put in a low-power state while still retaining memory contents; since the SDRAM will continue to refresh itself even without clocks from the microcontroller. Powerdown mode achieves even lower power, except the microcontroller must periodically wake up and issue refreshes if data retention is required.

Note that the EMIF module does not support Mobile SDRAM devices.

### 17.1.3 Functional Block Diagram

Figure 17-1 illustrates the connections between the EMIF and its internal requesters, along with the external EMIF pins. Section 17.2.2 contains a description of the entities internal to the SoC that can send requests to the EMIF, along with their prioritization. Section 17.2.3 describes the EMIF external pins and summarizes their purpose when interfacing with SDRAM and asynchronous devices.

Figure 17-1. EMIF Functional Block Diagram



## 17.2 EMIF Module Architecture

This section provides details about the architecture and operation of the EMIF. Both, SDRAM and asynchronous Interface are covered, along with other system-related issues such as clock control and pin multiplexing.

### 17.2.1 EMIF Clock Control

The EMIF clock is output on the EMIF\_CLK pin and should be used when interfacing to external SDRAM devices. The EMIF module gets the VCLK3 clock domain as the input. This clock domain is running at half the frequency of the main oscillator by default, that is, between 2.5MHz to 10MHz. The VCLK3 frequency is divided down from the HCLK domain frequency by a programmable divider (/1 to /16). Refer the Architecture chapter of the device technical reference manual for more information on configuring the VCLK3 domain frequency.

The EMIF clock (EMIF\_CLK) does not run during device reset. The application must clear the bit 8 of PINMMR29 register in the I/O Multiplexing Module to enable the EMIF\_CLK output to begin. Refer the I/O Multiplexing Module chapter of the TRM for more details.

### 17.2.2 EMIF Requests

Different sources within the SoC can make requests to the EMIF. These requests consist of accesses to SDRAM memory, asynchronous memory, and EMIF registers. The EMIF can process only one request at a time. Therefore a high performance crossbar switch exists within the SoC to provide prioritized requests from the different sources to the EMIF. The sources are:

1. CPU
2. DMA
3. Other master peripherals

If a request is submitted from two or more sources simultaneously, the crossbar switch will forward the highest priority request to the EMIF first. Upon completion of a request, the crossbar switch again evaluates the pending requests and forwards the highest priority pending request to the EMIF.

When the EMIF receives a request, it may or may not be immediately processed. In some cases, the EMIF will perform one or more auto refresh cycles before processing the request. For details on the EMIF's internal arbitration between performing requests and performing auto refresh cycles, see [Section 17.2.12](#).

### 17.2.3 EMIF Signal Descriptions

This section describes the function of each of the EMIF signals.

**Table 17-1. EMIF Pins Used to Access Both SDRAM and Asynchronous Memories**

Pin(s)	I/O	Description
EMIF_D[15:0]	I/O	<b>EMIF data bus.</b>
EMIF_A[21:0]	O	<b>EMIF address bus.</b> When interfacing to an SDRAM device, these pins are primarily used to provide the row and column address to the SDRAM. The mapping from the internal program address to the external values placed on these pins can be found in <a href="#">Table 17-13</a> . EMIF_A[10] is also used during the PRE command to select which banks to deactivate. When interfacing to an asynchronous device, these pins are used in conjunction with the EMIF_BA pins to form the address that is sent to the device. The mapping from the internal program address to the external values placed on these pins can be found in <a href="#">Section 17.2.5.1</a> .
EMIF_BA[1:0]	O	<b>EMIF bank address.</b> When interfacing to an SDRAM device, these pins are used to provide the bank address inputs to the SDRAM. The mapping from the internal program address to the external values placed on these pins can be found in <a href="#">Table 17-13</a> . When interfacing to an asynchronous device, these pins are used in conjunction with the EMIF_A pins to form the address that is sent to the device. The mapping from the internal program address to the external values placed on these pins can be found in <a href="#">Section 17.2.5.1</a> .
EMIF_nDQM[1:0]	O	<b>Active-low byte enables.</b> When interfacing to SDRAM, these pins are connected to the DQM pins of the SDRAM to individually enable/disable each of the bytes in a data access. When interfacing to an asynchronous device, these pins are connected to byte enables. See <a href="#">Section 17.2.5</a> for details.
EMIF_nWE	O	<b>Active-low write enable.</b> When interfacing to SDRAM, this pin is connected to the nWE pin of the SDRAM and is used to send commands to the device. When interfacing to an asynchronous device, this pin provides a signal which is active-low during the strobe period of an asynchronous write access cycle.

**Table 17-2. EMIF Pins Specific to SDRAM**

Pin(s)	I/O	Description
EMIF_nCS[0]	O	<b>Active-low chip enable pin for SDRAM devices.</b> This pin is connected to the chip-select pin of the attached SDRAM device and is used for enabling/disabling commands. By default, the EMIF keeps this SDRAM chip select active, even if the EMIF is not interfaced with an SDRAM device. This pin is deactivated when accessing the asynchronous memory bank and is reactivated on completion of the asynchronous access.
EMIF_nRAS	O	<b>Active-low row address strobe pin.</b> This pin is connected to the nRAS pin of the attached SDRAM device and is used for sending commands to the device.
EMIF_nCAS	O	<b>Active-low column address strobe pin.</b> This pin is connected to the nCAS pin of the attached SDRAM device and is used for sending commands to the device.
EMIF_CKE	O	<b>Clock enable pin.</b> This pin is connected to the CKE pin of the attached SDRAM device and is used for issuing the SELF REFRESH command which places the device in self refresh mode. See <a href="#">Section 17.2.4.7</a> for details.
EMIF_CLK	O	<b>SDRAM clock pin.</b> This pin is connected to the CLK pin of the attached SDRAM device. See <a href="#">Section 17.2.1</a> for details on the clock signal.

**Table 17-3. EMIF Pins Specific to Asynchronous Memory**

Pin(s)	I/O	Description
EMIF_nCS[4:2]	O	<b>Active-low chip enable pins for asynchronous devices.</b> These pins are meant to be connected to the chip-select pins of the attached asynchronous device. These pins are active only during accesses to the asynchronous memory.
EMIF_nWAIT	I	<b>Wait input with programmable polarity.</b> A connected asynchronous device can extend the strobe period of an access cycle by asserting the EMIF_nWAIT input to the EMIF as described in <a href="#">Section 17.2.5.6</a> . To enable this functionality, the EW bit in the asynchronous 1 configuration register (CE2CFG) must be set to 1. In addition, the WPO bit in CE2CFG must be configured to define the polarity of the EMIF_nWAIT pin.
EMIF_nOE	O	<b>Active-low pin enable for asynchronous devices.</b> This pin provides a signal which is active-low during the strobe period of an asynchronous read access cycle.
EMIF_RnW	O	<b>EMIF asynchronous read/write control.</b> This pin stays high during reads and stays low during writes (same duration as CS).

## 17.2.4 SDRAM Controller and Interface

The EMIF can gluelessly interface to most standard SDR SDRAM devices and supports such features as self refresh mode and prioritized refresh. In addition, it provides flexibility through programmable parameters such as the refresh rate, CAS latency, and many SDRAM timing parameters. The following sections include details on how to Interface and properly configure the EMIF to perform read and write operations to externally connected SDR SDRAM devices. Also, provides a detailed example of interfacing the EMIF to a common SDRAM device.

### 17.2.4.1 SDRAM Commands

The EMIF supports the SDRAM commands described in [Table 17-4](#). [Table 17-5](#) shows the truth table for the SDRAM commands, and an example timing waveform of the PRE command is shown in [Figure 17-2](#). EMIF\_A[10] is pulled low in this example to deactivate only the bank specified by the EMIF\_BA pins.

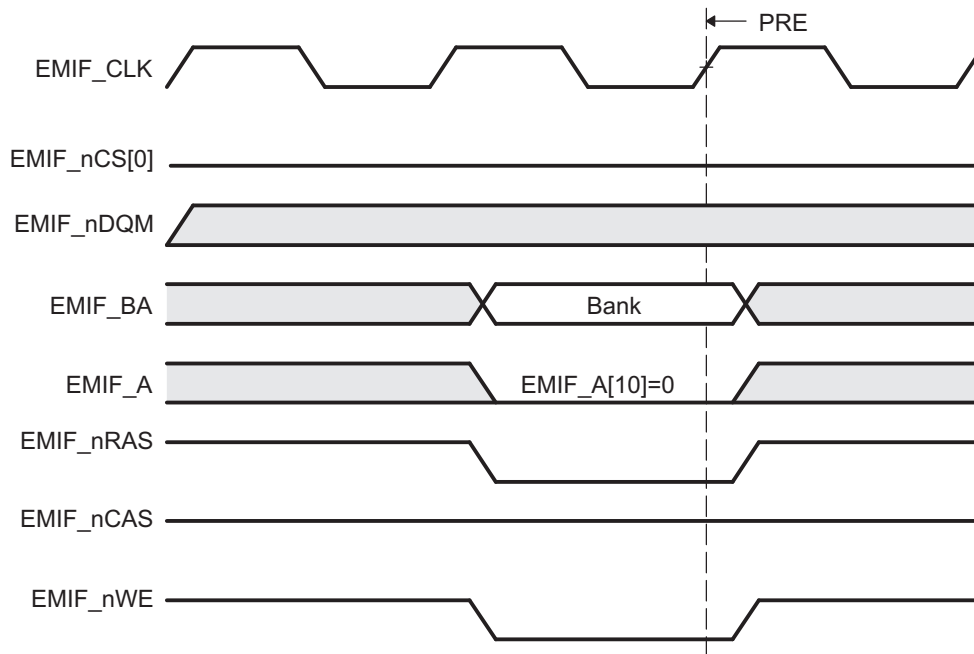
**Table 17-4. EMIF SDRAM Commands**

Command	Function
PRE	<b>Precharge.</b> Depending on the value of EMIF_A[10], the PRE command either deactivates the open row in all banks (EMIF_A[10] = 1) or only the bank specified by the EMIF_BA[1:0] pins (EMIF_A[10] = 0).
ACTV	<b>Activate.</b> The ACTV command activates the selected row in a particular bank for the current access.
READ	<b>Read.</b> The READ command outputs the starting column address and signals the SDRAM to begin the burst read operation. Address EMIF_A[10] is always pulled low to avoid auto precharge. This allows for better bank interleaving performance.
WRT	<b>Write.</b> The WRT command outputs the starting column address and signals the SDRAM to begin the burst write operation. Address EMIF_A[10] is always pulled low to avoid auto precharge. This allows for better bank interleaving performance.
BT	<b>Burst terminate.</b> The BT command is used to truncate the current read or write burst request.
LMR	<b>Load mode register.</b> The LMR command sets the mode register of the attached SDRAM devices and is only issued during the SDRAM initialization sequence described in <a href="#">Section 17.2.4.4</a> .
REFR	<b>Auto refresh.</b> The REFR command signals the SDRAM to perform an auto refresh according to its internal address.
SLFR	<b>Self refresh.</b> The self refresh command places the SDRAM into self refresh mode, during which it provides its own clock signal and auto refresh cycles.
NOP	<b>No operation.</b> The NOP command is issued during all cycles in which one of the above commands is not issued.

**Table 17-5. Truth Table for SDRAM Commands**

SDRAM Pins:	CKE	nCS	nRAS	nCAS	nWE	BA[1:0]	A[12:11]	A[10]	A[9:0]
EMIF Pins:	EMIF_CKE	EMIF_nCS[0]	EMIF_nRAS	EMIF_nCAS	EMIF_nWE	EMIF_BA[1:0]	EMIF_A[12:11]	EMIF_A[10]	EMIF_A[9:0]
PRE	H	L	L	H	L	Bank/X	X	L/H	X
ACTV	H	L	L	H	H	Bank	Row	Row	Row
READ	H	L	H	L	H	Bank	Column	L	Column
WRT	H	L	H	L	L	Bank	Column	L	Column
BT	H	L	H	H	L	X	X	X	X
LMR	H	L	L	L	L	X	Mode	Mode	Mode
REFR	H	L	L	L	H	X	X	X	X
SLFR	L	L	L	L	H	X	X	X	X
NOP	H	L	H	H	H	X	X	X	X

**Figure 17-2. Timing Waveform of SDRAM PRE Command**



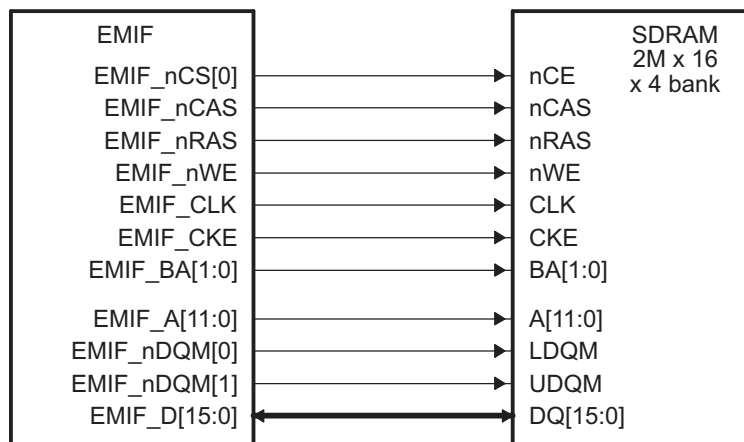
**17.2.4.2 Interfacing to SDRAM**

The EMIF supports a glueless interface to SDRAM devices with the following characteristics:

- Pre-charge bit is A[10]
- The number of column address bits is 8, 9, 10, or 11.
- The number of row address bits is 13, 14, 15, or 16.
- The number of internal banks is 1, 2, or 4.

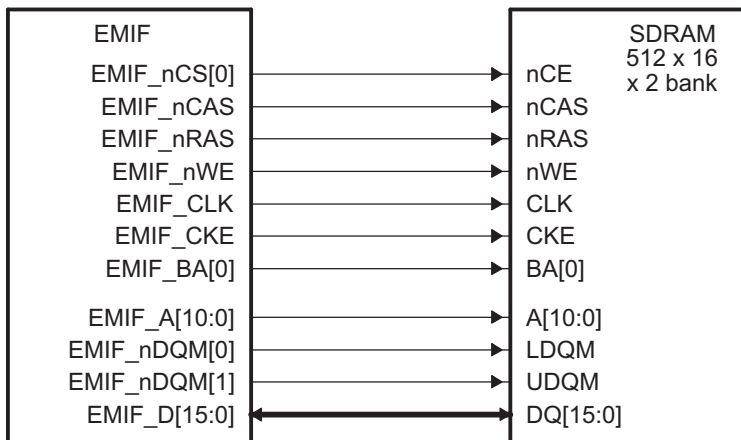
Figure 17-3 shows an interface between the EMIF and a 2M × 16 × 4 bank SDRAM device, and Figure 17-4 shows an interface between the EMIF and a 512K × 16 × 2 bank SDRAM device. For devices supporting 16-bit interface, refer to Table 17-6 for list of commonly-supported SDRAM devices and the required connections for the address pins.

**Figure 17-3. EMIF to 2M × 16 × 4 bank SDRAM Interface**





**Figure 17-4. EMIF to 512K × 16 × 2 bank SDRAM Interface**



**Table 17-6. 16-bit EMIF Address Pin Connections**

SDRAM Size	Width	Banks	Device	Address Pins
16M bits	×16	2	SDRAM	A[10:0]
			EMIF	EMIF_A[10:0]
64M bits	×16	4	SDRAM	A[11:0]
			EMIF	EMIF_A[11:0]
128M bits	×16	4	SDRAM	A[11:0]
			EMIF	EMIF_A[11:0]
256M bits	×16	4	SDRAM	A[12:0]
			EMIF	EMIF_A[12:0]
512M bits	×16	4	SDRAM	A[12:0]
			EMIF	EMIF_A[12:0]

### 17.2.4.3 SDRAM Configuration Registers

The operation of the EMIF's SDRAM interface is controlled by programming the appropriate configuration registers. This section describes the purpose and function of each configuration register, but [Section 17.3](#) should be referred for a more detailed description of each register, including the default registers values and bit-field positions. The following tables list the four such configuration registers, along with a description of each of their programmable fields.

---

**NOTE:** Writing to any of the fields: NM, CL, IBANK, and PAGESIZE in the SDRAM configuration register (SDCR) causes the EMIF to abandon whatever it is currently doing and trigger the SDRAM initialization procedure described in [Section 17.2.4.4](#).

---

**Table 17-7. Description of the SDRAM Configuration Register (SDCR)**

Parameter	Description
SR	This bit controls entering and exiting of the Self-Refresh mode. The field should be written using a byte-write to the upper byte of SDCR to avoid triggering the SDRAM initialization sequence.
PD	This bit controls entering and exiting of the Power down mode. The field should be written using a byte-write to the upper byte of SDCR to avoid triggering the SDRAM initialization sequence. If both SR and PD bits are set, the EMIF will go into Self Refresh.
PDWR	Perform refreshes during Power Down. Writing a 1 to this bit will cause the EMIF to exit the power down state and issue an AUTO REFRESH command every time Refresh May level is set. The field should be written using a byte-write to the upper byte of SDCR to avoid triggering the SDRAM initialization sequence. This bit should be set along with PD when entering power-down mode.
NM	<b>Narrow Mode.</b> This bit defines the width of the data bus between the EMIF and the attached SDRAM device. When set to 1, the data bus is set to 16-bits. When set to 0, the data bus is set to 32-bits. This bit must always be set to 1.
CL	<b>CAS latency.</b> This field defines the number of clock cycles between when an SDRAM issues a READ command and when the first piece of data appears on the bus. The value in this field is sent to the attached SDRAM device via the LOAD MODE REGISTER command during the SDRAM initialization procedure as described in <a href="#">Section 17.2.4.4</a> . Only, values of 2h (CAS latency = 2) and 3h (CAS latency = 3) are supported and should be written to this field. A 1 must be simultaneously written to the BIT11_9LOCK bit field of SDCR in order to write to the CL bit field.
IBANK	<b>Number of Internal SDRAM Banks.</b> This field defines the number of banks inside the attached SDRAM devices in the following way: <ul style="list-style-type: none"> <li>• When IBANK = 0, 1 internal bank is used</li> <li>• When IBANK = 1h, 2 internal banks are used</li> <li>• When IBANK = 2h, 4 internal banks are used</li> </ul> This field value affects the mapping of logical addresses to SDRAM row, column, and bank addresses. See <a href="#">Section 17.2.4.11</a> for details.
PAGESIZE	<b>Page Size.</b> This field defines the internal page size of the attached SDRAM devices in the following way: <ul style="list-style-type: none"> <li>• When PAGESIZE = 0, 256-word pages are used</li> <li>• When PAGESIZE = 1h, 512-word pages are used</li> <li>• When PAGESIZE = 2h, 1024-word pages are used</li> <li>• When PAGESIZE = 3h, 2048-word pages are used</li> </ul> This field value affects the mapping of logical addresses to SDRAM row, column, and bank addresses. See <a href="#">Section 17.2.4.11</a> for details.

**Table 17-8. Description of the SDRAM Refresh Control Register (SDRCR)**

Parameter	Description
RR	<b>Refresh Rate.</b> This field controls the rate at which attached SDRAM devices will be refreshed. The following equation can be used to determine the required value of RR for an SDRAM device: <ul style="list-style-type: none"> <li>• <math>RR = f_{EMIF\_CLK} / (\text{Required SDRAM Refresh Rate})</math></li> </ul> More information about the operation of the SDRAM refresh controller can be found in <a href="#">Section 17.2.4.6</a> .

**Table 17-9. Description of the SDRAM Timing Register (SDTIMR)**

Parameter	Description
T_RFC	<b>SDRAM Timing Parameters.</b> These fields configure the EMIF to comply with the AC timing requirements of the attached SDRAM devices. This allows the EMIF to avoid violating SDRAM timing constraints and to more efficiently schedule its operations. More details about each of these parameters can be found in the register description in <a href="#">Section 17.3.6</a> . These parameters should be set to satisfy the corresponding timing requirements found in the SDRAM's datasheet.
T_RP	
T_RCD	
T_WR	
T_RAS	
T_RC	
T_RRD	

**Table 17-10. Description of the SDRAM Self Refresh Exit Timing Register (SDSRETR)**

Parameter	Description
T_XS	<b>Self Refresh Exit Parameter.</b> The T_XS field of this register informs the EMIF about the minimum number of EMIF_CLK cycles required between exiting Self Refresh and issuing any command. This parameter should be set to satisfy the $t_{XSR}$ value for the attached SDRAM device.

#### 17.2.4.4 SDRAM Auto-Initialization Sequence

The EMIF automatically performs an SDRAM initialization sequence, regardless of whether it is interfaced to an SDRAM device, when either of the following two events occur:

- The EMIF comes out of reset. No memory accesses to the SDRAM and Asynchronous interfaces are performed until this auto-initialization is complete.
- A write is performed to any of the three least significant bytes of the SDRAM configuration register (SDCR)

An SDRAM initialization sequence consists of the following steps:

1. If the initialization sequence is activated by a write to SDCR, and if any of the SDRAM banks are open, the EMIF issues a PRE command with EMIF\_A[10] held high to indicate all banks. This is done so that the maximum ACTV to PRE timing for an SDRAM is not violated.
2. The EMIF drives EMIF\_CKE high and begins continuously issuing NOP commands until eight SDRAM refresh intervals have elapsed. An SDRAM refresh interval is equal to the value of the RR field of SDRAM refresh control register (SDRCR), divided by the frequency of EMIF\_CLK ( $RR/f_{EMIF\_CLK}$ ). This step is used to avoid violating the Power-up constraint of most SDRAM devices that requires 200  $\mu$ s (sometimes 100  $\mu$ s) between receiving stable Vdd and CLK and the issuing of a PRE command. Depending on the frequency of EMIF\_CLK, this step may or may not be sufficient to avoid violating the SDRAM constraint. See [Section 17.2.4.5](#) for more information.
3. After the refresh intervals have elapsed, the EMIF issues a PRE command with EMIF\_A[10] held high to indicate all banks.
4. The EMIF issues eight AUTO REFRESH commands.
5. The EMIF issues the LMR command with the EMIF\_A[9:0] pins set as described in [Table 17-11](#).
6. Finally, the EMIF performs a refresh cycle, which consists of the following steps:
  - (a) Issuing a PRE command with EMIF\_A[10] held high if any banks are open
  - (b) Issuing an REF command

**Table 17-11. SDRAM LOAD MODE REGISTER Command**

EMIF_A[9:7]	EMIF_A[6:4]	EMIF_A[3]	EMIF_A[2:0]
0 (Write bursts are of the programmed burst length in EMIF_A[2:0])	These bits control the CAS latency of the SDRAM and are set according to CL field in the SDRAM configuration register (SDCR) as follows: <ul style="list-style-type: none"> <li>• If CL = 2, EMIF_A[6:4] = 2h (CAS latency = 2)</li> <li>• If CL = 3, EMIF_A[6:4] = 3h (CAS latency = 3)</li> </ul>	0 (Sequential Burst Type. Interleaved Burst Type not supported)	These bits control the burst length of the SDRAM and are set according to the NM field in the SDRAM configuration register (SDCR) as follows: <ul style="list-style-type: none"> <li>• If NM = 0, EMIF_A[2:0] = 2h (Burst Length = 4)</li> <li>• If NM = 1, EMIF_A[2:0] = 3h (Burst Length = 8)</li> </ul>

#### 17.2.4.5 SDRAM Configuration Procedure

There are two different SDRAM configuration procedures. Although EMIF automatically performs the SDRAM initialization sequence described in [Section 17.2.4.4](#) when coming out of reset, it is recommended to follow one of the procedures listed below before performing any EMIF memory requests. Procedure A should be followed if it is determined that the SDRAM Power-up constraint was not violated during the SDRAM Auto-Initialization Sequence detailed in [Section 17.2.4.4](#) on coming out of Reset. The SDRAM Power-up constraint specifies that 200  $\mu$ s (sometimes 100  $\mu$ s) should exist between receiving stable V<sub>dd</sub> and CLK and the issuing of a PRE command. Procedure B should be followed if the SDRAM Power-up constraint was violated. The 200  $\mu$ s (100  $\mu$ s) SDRAM Power-up constraint will be violated if the frequency of EMIF\_CLK is greater than 50 MHz (100 MHz for 100  $\mu$ s SDRAM power-up constraint) during SDRAM Auto-Initialization Sequence. Procedure B should be followed if there is any doubt that the Power-up constraint was not met.

**Procedure A** — Following is the procedure to be followed if the SDRAM Power-up constraint was NOT violated:

1. Place the SDRAM into Self-Refresh Mode by setting the SR bit of SDCR to 1. A byte-write to the upper byte of SDCR should be used to avoid restarting the SDRAM Auto-Initialization Sequence described in [Section 17.2.4.4](#). The SDRAM should be placed into Self-Refresh mode when changing the frequency of EMIF\_CLK to avoid incurring the 200  $\mu$ s Power-up constraint again.
2. Configure the desired EMIF\_CLK clock frequency. The frequency of the memory clock must meet the timing requirements in the SDRAM manufacturer's documentation and the timing limitations shown in the electrical specifications of the device datasheet.
3. Remove the SDRAM from Self-Refresh Mode by clearing the SR bit of SDCR to 0. A byte-write to the upper byte of SDCR should be used to avoid restarting the SDRAM Auto-Initialization Sequence described in [Section 17.2.4.4](#).
4. Program SDTIMR and SDSRETR to satisfy the timing requirements for the attached SDRAM device. The timing parameters should be taken from the SDRAM datasheet.
5. Program the RR field of SDRCR to match that of the attached device's refresh interval. See [Section 17.2.4.6.1](#) details on determining the appropriate value.
6. Program SDCR to match the characteristics of the attached SDRAM device. This will cause the auto-initialization sequence in [Section 17.2.4.4](#) to be re-run. This second initialization generally takes much less time due to the increased frequency of EMIF\_CLK.

**Procedure B** — Following is the procedure to be followed if the SDRAM Power-up constraint was violated:

1. Configure the desired EMIF\_CLK clock frequency. The frequency of the memory clock must meet the timing requirements in the SDRAM manufacturer's documentation and the timing limitations shown in the electrical specifications of the device datasheet.
2. Program SDTIMR and SDSRETR to satisfy the timing requirements for the attached SDRAM device. The timing parameters should be taken from the SDRAM datasheet.
3. Program the RR field of SDRCR such that the following equation is satisfied:  $(RR \times 8)/(f_{EMIF\_CLK}) > 200 \mu s$  (sometimes 100  $\mu$ s). For example, an EMIF\_CLK frequency of 100 MHz would require setting RR to 2501 (9C5h) or higher to meet a 200  $\mu$ s constraint.

4. Program SDCR to match the characteristics of the attached SDRAM device. This will cause the auto-initialization sequence in [Section 17.2.4.4](#) to be re-run with the new value of RR.
5. Perform a read from the SDRAM to assure that step 5 of this procedure will occur after the initialization process has completed. Alternatively, wait for 200  $\mu$ s instead of performing a read.
6. Finally, program the RR field to match that of the attached device's refresh interval. See [Section 17.2.4.6.1](#) details on determining the appropriate value.

After following the above procedure, the EMIF is ready to perform accesses to the attached SDRAM device. See for an example of configuring the SDRAM interface.

#### 17.2.4.6 EMIF Refresh Controller

An SDRAM device requires that each of its rows be refreshed at a minimum required rate. The EMIF can meet this constraint by performing auto refresh cycles at or above this required rate. An auto refresh cycle consists of issuing a PRE command to all banks of the SDRAM device followed by issuing a REFR command. To inform the EMIF of the required rate for performing auto refresh cycles, the RR field of the SDRAM refresh control register (SDRCR) must be programmed. The EMIF will use this value along with two internal counters to automatically perform auto refresh cycles at the required rate. The auto refresh cycles cannot be disabled, even if the EMIF is not interfaced with an SDRAM. The remainder of this section details the EMIF's refresh scheme and provides an example for determining the appropriate value to place in the RR field of SDRCR.

The two counters used to perform auto-refresh cycles are a 13-bit refresh interval counter and a 4-bit refresh backlog counter. At reset and upon writing to the RR field, the refresh interval counter is loaded with the value from RR field and begins decrementing, by one, each EMIF clock cycle. When the refresh interval counter reaches zero, the following actions occur:

- The refresh interval counter is reloaded with the value from the RR field and restarts decrementing.
- The 4-bit refresh backlog counter increments unless it has already reached its maximum value.

The refresh backlog counter records the number of auto refresh cycles that the EMIF currently has outstanding. This counter is decremented by one each time an auto refresh cycle is performed and incremented by one each time the refresh interval counter expires. The refresh backlog counter saturates at the values of 0000b and 1111b. The EMIF uses the refresh backlog counter to determine the urgency with which an auto refresh cycle should be performed. The four levels of urgency are described in [Table 17-12](#). This refresh scheme allows the required refreshes to be performed with minimal impact on access requests.

**Table 17-12. Refresh Urgency Levels**

Urgency Level	Refresh Backlog Counter Range	Action Taken
Refresh May	1-3	An auto-refresh cycle is performed only if the EMIF has no requests pending and none of the SDRAM banks are open.
Refresh Release	4-7	An auto-refresh cycle is performed if the EMIF has no requests pending, regardless of whether any SDRAM banks are open.
Refresh Need	8-11	An auto-refresh cycle is performed at the completion of the current access unless there are read requests pending.
Refresh Must	12-15	Multiple auto-refresh cycles are performed at the completion of the current access until the Refresh Release urgency level is reached. At that point, the EMIF can begin servicing any new read or write requests.

### 17.2.4.6.1 Determining the Appropriate Value for the RR Field

The value that should be programmed into the RR field of SDCR can be calculated by using the frequency of the EMIF\_CLK signal ( $f_{EMIF\_CLK}$ ) and the required refresh rate of the SDRAM ( $f_{Refresh}$ ). The following formula can be used:

$$RR = f_{EMIF\_CLK} / f_{Refresh}$$

The SDRAM datasheet often communicates the required SDRAM Refresh Rate in terms of the number of REFR commands required in a given time interval. The required SDRAM Refresh Rate in the formula above can therefore be calculated by dividing the number of required cycles per time interval ( $n_{cycles}$ ) by the time interval given in the datasheet ( $t_{Refresh\ Period}$ ):

$$f_{Refresh} = n_{cycles} / t_{Refresh\ Period}$$

Combining these formulas, the value that should be programmed into the RR field can be computed as:

$$RR = f_{EMIF\_CLK} \times t_{Refresh\ Period} / n_{cycles}$$

The following example illustrates calculating the value of RR. Given that:

- $f_{EMIF\_CLK} = 100\text{ MHz}$  (frequency of the EMIF clock)
- $t_{Refresh\ Period} = 64\text{ ms}$  (required refresh interval of the SDRAM)
- $n_{cycles} = 8192$  (number of cycles in a refresh interval for the SDRAM)

RR can be calculated as:

$$RR = 100\text{ MHz} \times 64\text{ ms} / 8192$$

$$RR = 781.25$$

$$RR = 782\text{ cycles} = 30\text{Eh cycles}$$

### 17.2.4.7 Self-Refresh Mode

The EMIF can be programmed to enter the self-refresh state by setting the SR bit of SDCR to 1. This will cause the EMIF to issue the SLFR command after completing any outstanding SDRAM access requests and clearing the refresh backlog counter by performing one or more auto refresh cycles. This places the attached SDRAM device into self-refresh mode in which it consumes a minimal amount of power while performing its own refresh cycles. The SR bit should be set and cleared using a byte-write to the upper byte of the SDRAM configuration register (SDCR) to avoid triggering the SDRAM initialization sequence.

While in the self-refresh state, the EMIF continues to service asynchronous bank requests and register accesses as normal, with one caveat. The EMIF will not park the data bus following a read to asynchronous memory while in the self-refresh state. Instead, the EMIF tri-states the data bus. Therefore, it is not recommended to perform asynchronous read operations while the EMIF is in the self-refresh state, in order to prevent floating inputs on the data bus. More information about data bus parking can be found in [Section 17.2.6](#).

The EMIF will exit from the self-refresh state if either of the following events occur:

- The SR bit of SDCR is cleared to 0.
- An SDRAM accesses is requested.

The EMIF exits from the self-refresh state by driving EMIF\_CKE high and performing an auto refresh cycle.

The attached SDRAM device should also be placed into Self-Refresh Mode when changing the frequency of EMIF\_CLK. If the frequency of EMIF\_CLK changes while the SDRAM is not in Self-Refresh Mode, Procedure B in [Section 17.2.4.5](#) should be followed to reinitialize the device.

### 17.2.4.8 Power Down Mode

To support low-power modes, the EMIF can be requested to issue a POWER DOWN command to the SDRAM by setting the PD bit in the SDRAM configuration register (SDCR). When this bit is set, the EMIF will continue normal operation until all outstanding memory access requests have been serviced and the SDRAM refresh backlog (if there is one) has been cleared. At this point the EMIF will enter the power-down state. Upon entering this state, the EMIF will issue a POWER DOWN command (same as a NOP command but driving EMIF\_CKE low on the same cycle). The EMIF then maintains EMIF\_CKE low until it exits the power-down state.

Since the EMIF services the refresh backlog before it enters the power-down state, all internal banks of the SDRAM are closed (precharged) prior to issuing the POWER DOWN command. Therefore, the EMIF only supports Precharge Power Down. The EMIF does not support Active Power Down, where internal banks of the SDRAM are open (active) before the POWER DOWN command is issued.

During the power-down state, the EMIF services the SDRAM, asynchronous memory, and register accesses as normal, returning to the power-down state upon completion.

The PDWR bit in SDCR indicates whether the EMIF should perform refreshes in power-down state. If the PDWR bit is set, the EMIF exits the power-down state every time the Refresh Must level is set, performs AUTO REFRESH commands to the SDRAM, and returns back to the power-down state. This evenly distributes the refreshes to the SDRAM in power-down state. If the PDWR bit is not set, the EMIF does not perform any refreshes to the SDRAM. Therefore, the data integrity of the SDRAM is not assured upon power down exit if the PDWR bit is not set.

If the PD bit is cleared while in the power-down state, the EMIF will come out of the power-down state. The EMIF:

- Drives EMIF\_CKE high.
- Enters its idle state.



### 17.2.4.9 SDRAM Read Operation

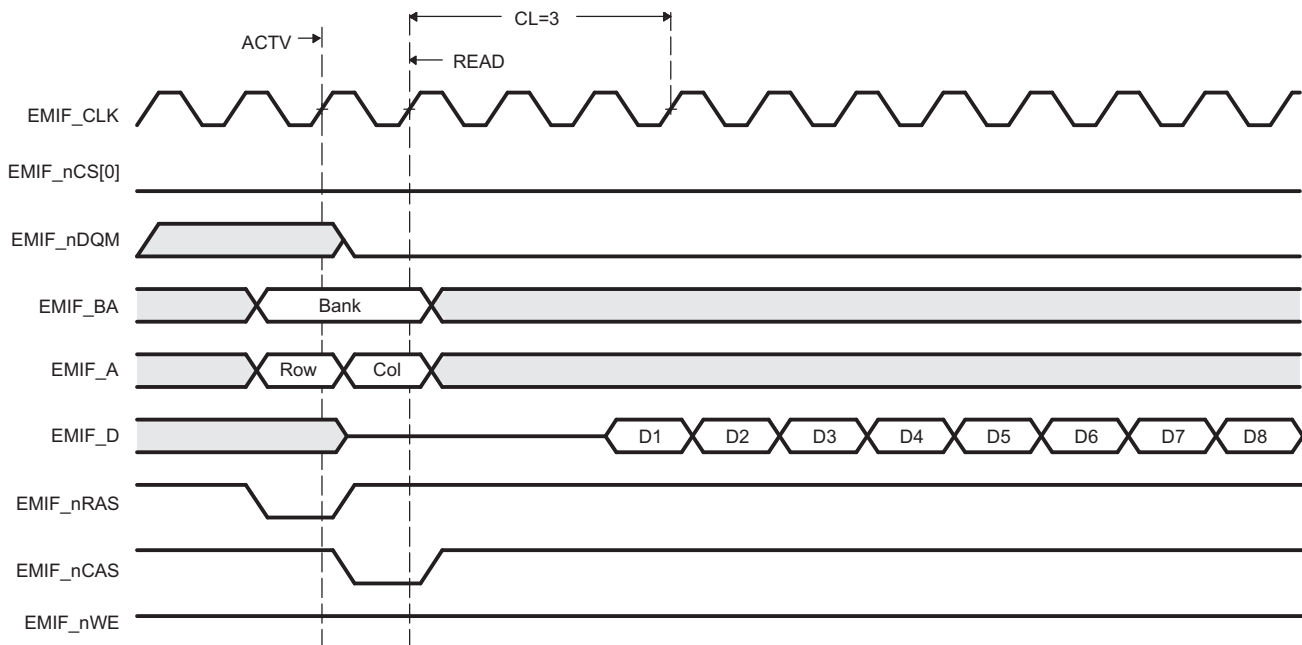
When the EMIF receives a read request to SDRAM from one of the requesters listed in [Section 17.2.2](#), it performs one or more read access cycles. A read access cycle begins with the issuing of the ACTV command to select the desired bank and row of the SDRAM device. After the row has been opened, the EMIF proceeds to issue a READ command while specifying the desired bank and column address. EMIF\_A[10] is held low during the READ command to avoid auto-precharging. The READ command signals the SDRAM device to start bursting data from the specified address while the EMIF issues NOP commands. Following a READ command, the CL field of the SDRAM configuration register (SDCR) defines how many delay cycles will be present before the read data appears on the data bus. This is referred to as the CAS latency.

[Figure 17-5](#) shows the signal waveforms for a basic SDRAM read operation in which a burst of data is read from a single page. When the EMIF SDRAM interface is configured to 16 bit by setting the NM bit of the SDRAM configuration register (SDCR) to 1, a burst size of eight is used. [Figure 17-5](#) shows a burst size of eight.

The EMIF will truncate a series of bursting data if the remaining addresses of the burst are not required to complete the request. The EMIF can truncate the burst in three ways:

- By issuing another READ to the same page in the same bank.
- By issuing a PRE command in order to prepare for accessing a different page of the same bank.
- By issuing a BT command in order to prepare for accessing a page in a different bank.

**Figure 17-5. Timing Waveform for Basic SDRAM Read Operation**



Several other pins are also active during a read access. The EMIF\_nDQM[1:0] pins are driven low during the READ commands and are kept low during the NOP commands that correspond to the burst request. The state of the other EMIF pins during each command can be found in [Table 17-5](#).

The EMIF schedules its commands based on the timing information that is provided to it in the SDRAM timing register (SDTIMR). The values for the timing parameters in this register should be chosen to satisfy the timing requirements listed in the SDRAM datasheet. The EMIF uses this timing information to avoid violating any timing constraints related to issuing commands. This is commonly accomplished by inserting NOP commands between various commands during an access. Refer to the register description of SDTIMR in [Section 17.3.6](#) for more details on the various timing parameters.

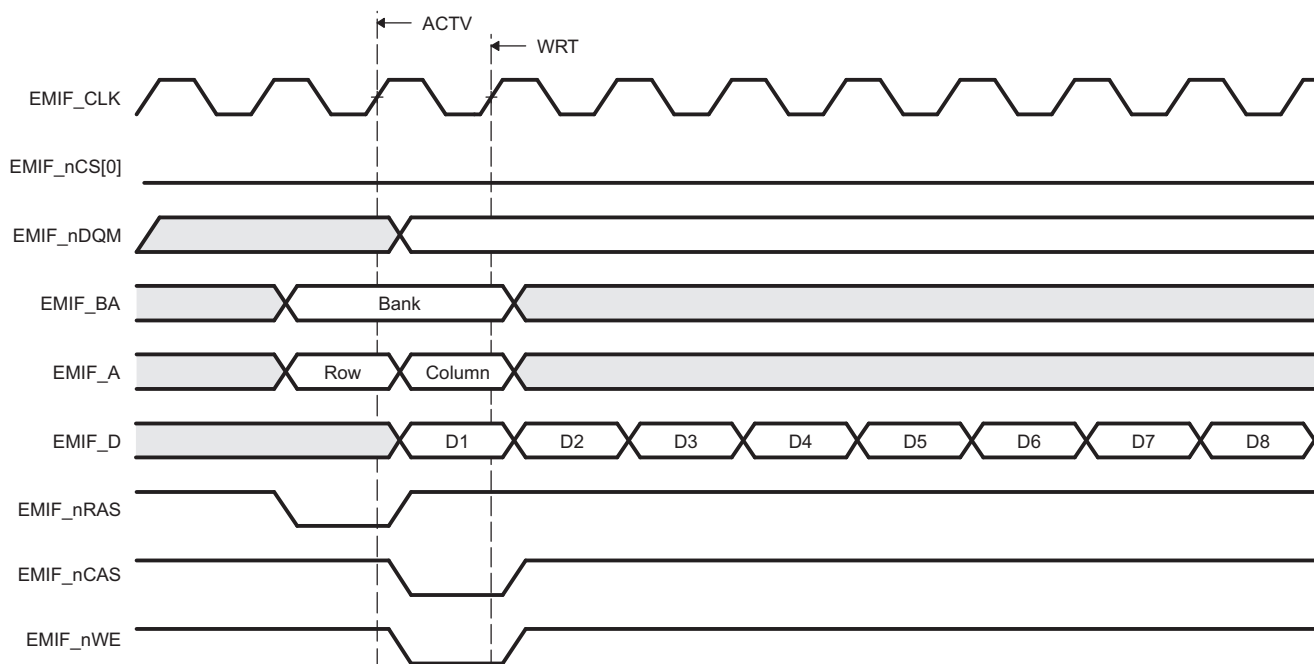


### 17.2.4.10 SDRAM Write Operations

When the EMIF receives a write request to SDRAM from one of the requesters listed in [Section 17.2.2](#), it performs one or more write-access cycles. A write-access cycle begins with the issuing of the ACTV command to select the desired bank and row of the SDRAM device. After the row has been opened, the EMIF proceeds to issue a WRT command while specifying the desired bank and column address. EMIF\_A[10] is held low during the WRT command to avoid auto-precharging. The WRT command signals the SDRAM device to start writing a burst of data to the specified address while the EMIF issues NOP commands. The associated write data will be placed on the data bus in the cycle concurrent with the WRT command and with subsequent burst continuation NOP commands.

[Figure 17-6](#) shows the signal waveforms for a basic SDRAM write operation in which a burst of data is read from a single page. When the EMIF SDRAM interface is configured to 16-bit by setting the NM bit of the SDRAM configuration register (SDCR) to 1, a burst size of eight is used. [Figure 17-6](#) shows a burst size of eight.

**Figure 17-6. Timing Waveform for Basic SDRAM Write Operation**



The EMIF will truncate a series of bursting data if the remaining addresses of the burst are not part of the write request. The EMIF can truncate the burst in three ways:

- By issuing another WRT to the same page
- By issuing a PRE command in order to prepare for accessing a different page of the same bank
- By issuing a BT command in order to prepare for accessing a page in a different bank

Several other pins are also active during a write access. The EMIF\_nDQM[1:0] pins are driven to select which bytes of the data word will be written to the SDRAM device. They are also used to mask out entire undesired data words during a burst access. The state of the other EMIF pins during each command can be found in [Table 17-5](#).

The EMIF schedules its commands based on the timing information that is provided to it in the SDRAM timing register (SDTIMR). The values for the timing parameters in this register should be chosen to satisfy the timing requirements listed in the SDRAM datasheet. The EMIF uses this timing information to avoid violating any timing constraints related to issuing commands. This is commonly accomplished by inserting NOP commands during various cycles of an access. Refer to the register description of SDTIMR in [Section 17.3.6](#) for more details on the various timing parameters.

### 17.2.4.11 Mapping from Logical Address to EMIF Pins

When the EMIF receives an SDRAM access request, it must convert the address of the access into the appropriate signals to send to the SDRAM device. The details of this address mapping are shown in [Table 17-13](#) for 16-bit operation. Using the settings of the IBANK and PAGESIZE fields of the SDRAM configuration register (SDCR), the EMIF determines which bits of the logical address will be mapped to the SDRAM row, column, and bank addresses.

As the logical address is incremented by one halfword (16-bit operation), the column address is likewise incremented by one until a page boundary is reached. When the logical address increments across a page boundary, the EMIF moves into the same page in the next bank of the attached device by incrementing the bank address EMIF\_BA and resetting the column address. The page in the previous bank is left open until it is necessary to close it. This method of traversal through the SDRAM banks helps maximize the number of open banks inside of the SDRAM and results in an efficient use of the device. There is no limitation on the number of banks that can be open at one time, but only one page within a bank can be open at a time.

The EMIF uses the EMIF\_nDQM[1:0] pins during a WRT command to mask out selected bytes or entire words. The EMIF\_nDQM[1:0] pins are always low during a READ command.

**Table 17-13. Mapping from Logical Address to EMIF Pins for 16-bit SDRAM**

IBANK	PAGESIZE	Logical Address												
		31:27	26	25	24	23	22	21:14	13	12	11	10	9	8:1
0	0	Row Address										Col Address	EMIF_nDQM[0]	
1	0	Row Address										EMIF_BA[0]	Col Address	EMIF_nDQM[0]
2	0	Row Address										EMIF_BA[1:0]	Col Address	EMIF_nDQM[0]
0	1	Row Address										Column Address		EMIF_nDQM[0]
1	1	Row Address										EMIF_BA[0]	Column Address	EMIF_nDQM[0]
2	1	Row Address										EMIF_BA[1:0]	Column Address	EMIF_nDQM[0]
0	2	Row Address										Column Address		EMIF_nDQM[0]
1	2	Row Address										EMIF_BA[0]	Column Address	EMIF_nDQM[0]
2	2	Row Address										EMIF_BA[1:0]	Column Address	EMIF_nDQM[0]
0	3	Row Address										Column Address		EMIF_nDQM[0]
1	3	Row Address										EMIF_BA[0]	Column Address	EMIF_nDQM[0]
2	3	Row Address										EMIF_BA[1:0]	Column Address	EMIF_nDQM[0]

---

**NOTE:** The upper bit of the Row Address is used only when addressing 256-Mbit and 512-Mbit SDRAM memories.

---

### 17.2.5 Asynchronous Controller and Interface

The EMIF easily interfaces to a variety of asynchronous devices including NOR Flash, NAND Flash, and SRAM. It can be operated in two major modes (see [Table 17-14](#)):

- Normal Mode
- Select Strobe Mode

**Table 17-14. Normal Mode vs. Select Strobe Mode**

Mode	Function of EMIF_nDQM pins	Operation of EMIF_nCS[4:2]
Normal Mode	Byte enables	Active during the entire asynchronous access cycle
Select Strobe Mode	Byte enables	Active only during the strobe period of an access cycle

The first mode of operation is Normal Mode, in which the EMIF\_nDQM pins of the EMIF function as byte enables. In this mode, the EMIF\_nCS[4:2] pins behaves as typical chip select signals, remaining active for the duration of the asynchronous access. See [Section 17.2.5.1](#) for an example interface with multiple 8-bit devices.

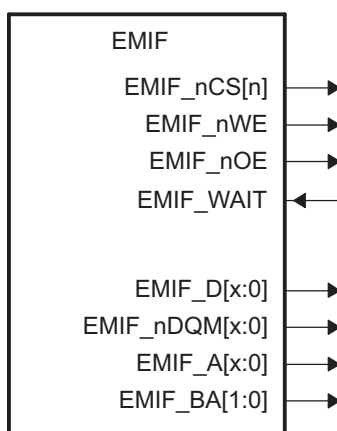
The second mode of operation is Select Strobe Mode, in which the EMIF\_nCS[4:2] pins act as a strobe, active only during the strobe period of an access. In this mode, the EMIF\_nDQM pins of the EMIF function as standard byte enables for reads and writes. A summary of the differences between the two modes of operation are shown in [Table 17-14](#). Refer to [Section 17.2.5.4](#) for the details of asynchronous operations in Normal Mode, and to [Section 17.2.5.5](#) for the details of asynchronous operations in Select Strobe Mode. The EMIF hardware defaults to Normal Mode, but can be manually switched to Select Strobe Mode by setting the SS bit in the asynchronous  $m$  ( $m = 1, 2, 3, \text{ or } 4$ ) configuration register (CE $n$ CFG) ( $n = 2, 3, \text{ or } 4$ ). Throughout the chapter,  $m$  can hold the values 1, 2, 3 or 4; and  $n$  can hold the values 2, 3, or 4.

The EMIF also provides configurable cycle timing parameters and an Extended Wait Mode that allows the connected device to extend the strobe period of an access cycle. The following sections describe the features related to interfacing with external asynchronous devices.

#### 17.2.5.1 Interfacing to Asynchronous Memory

[Figure 17-7](#) shows the EMIF's external pins used in interfacing with an asynchronous device. In EMIF\_nCS[ $n$ ],  $n = 2, 3, \text{ or } 4$ .

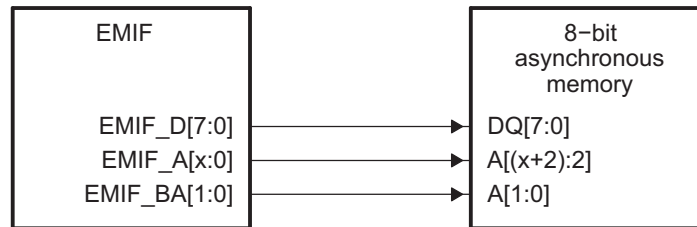
**Figure 17-7. EMIF Asynchronous Interface**



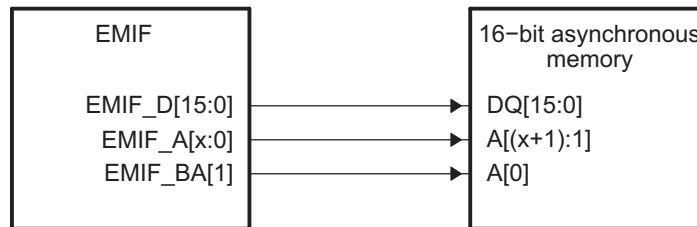
Of special note is the connection between the EMIF and the external device's address bus. The EMIF address pin EMIF\_A[0] always provides the least significant bit of a 32-bit word address. Therefore, when interfacing to a 16-bit or 8-bit asynchronous device, the EMIF\_BA[1] and EMIF\_BA[0] pins provide the least-significant bits of the halfword or byte address, respectively. Additionally, when the EMIF interfaces to a 16-bit asynchronous device, the EMIF\_BA[0] pin can serve as the upper address line EMIF\_A[22]. Figure 17-8 and Figure 17-9 show the mapping between the EMIF and the connected device's data and address pins for various programmed data bus widths. The data bus width may be configured in the asynchronous *n* configuration register (CE<sub>n</sub>CFG).

Figure 17-9 shows a common interface between the EMIF and external asynchronous memory. Figure 17-9 shows an interface between the EMIF and an external memory with byte enables. The EMIF should be operated in either Normal Mode or Select Strobe Mode when using this interface, so that the EMIF\_nDQM signals operate as byte enables.

**Figure 17-8. EMIF to 8-bit/16-bit Memory Interface**

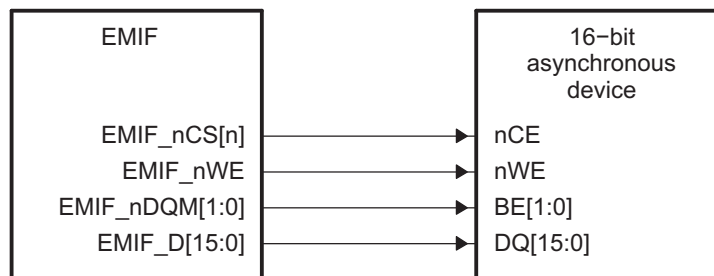


a) EMIF to 8-bit memory interface



b) EMIF to 16-bit memory interface

**Figure 17-9. Common Asynchronous Interface**



### 17.2.5.2 Accessing Larger Asynchronous Memories

The device has 22 dedicated EMIF address lines. If a device such as a large asynchronous flash needs to be attached to the EMIF, then GPIO pins may be used to control the flash device's upper address lines.

### 17.2.5.3 Configuring the EMIF for Asynchronous Accesses

The operation of the EMIF's asynchronous interface can be configured by programming the appropriate register fields. The reset value and bit position for each register field can be found in [Section 17.3](#). The following tables list the register fields that can be programmed and describe the purpose of each field. These registers can be programmed prior to accessing the external memory, and the transfer following a write to these registers will use the new configuration.

**Table 17-15. Description of the Asynchronous *m* Configuration Register (CE<sub>*n*</sub>CFG)**

Parameter	Description
SS	<p><b>Select Strobe mode.</b> This bit selects the EMIF's mode of operation in the following way:</p> <ul style="list-style-type: none"> <li>• SS = 0 selects Normal Mode <ul style="list-style-type: none"> <li>– EMIF_nDQM pins function as byte enables</li> <li>– EMIF_nCS[4:2] active for duration of access</li> </ul> </li> <li>• SS = 1 selects Select Strobe Mode <ul style="list-style-type: none"> <li>– EMIF_nDQM pins function as byte enables</li> <li>– EMIF_nCS[4:2] acts as a strobe.</li> </ul> </li> </ul>
EW	<p><b>Extended Wait Mode enable.</b></p> <ul style="list-style-type: none"> <li>• EW = 0 disables Extended Wait Mode</li> <li>• EW = 1 enables Extended Wait Mode</li> </ul> <p>When set to 1, the EMIF enables its Extended Wait Mode in which the strobe width of an access cycle can be extended in response to the assertion of the EMIF_nWAIT pin. The WP<sub><i>n</i></sub> bit in the asynchronous wait cycle configuration register (AWCC) controls to polarity of EMIF_nWAIT pin. See <a href="#">Section 17.2.5.6</a> for more details on this mode of operation.</p>
W_SETUP/R_SETUP	<p><b>Read/Write setup widths.</b></p> <p>These fields define the number of EMIF clock cycles of setup time for the address pins (EMIF_A and EMIF_BA), byte enables (EMIF_nDQM), and asynchronous chip enable (EMIF_nCS[4:2]) before the read strobe pin (EMIF_nOE) or write strobe pin (EMIF_nWE) falls, minus one cycle. For writes, the W_SETUP field also defines the setup time for the data pins (EMIF_D). Refer to the datasheet of the external asynchronous device to determine the appropriate setting for this field.</p>
W_STROBE/R_STROBE	<p><b>Read/Write strobe widths.</b></p> <p>These fields define the number of EMIF clock cycles between the falling and rising of the read strobe pin (EMIF_nOE) or write strobe pin (EMIF_nWE), minus one cycle. If Extended Wait Mode is enabled by setting the EW field in the asynchronous <i>n</i> configuration register (CE<sub><i>n</i></sub>CFG), these fields must be set to a value greater than zero. Refer to the datasheet of the external asynchronous device to determine the appropriate setting for this field.</p>
W_HOLD/R_HOLD	<p><b>Read/Write hold widths.</b></p> <p>These fields define the number of EMIF clock cycles of hold time for the address pins (EMIF_A and EMIF_BA), byte enables (EMIF_nDQM), and asynchronous chip enable (EMIF_nCS[4:2]) after the read strobe pin (EMIF_nOE) or write strobe pin (EMIF_nWE) rises, minus one cycle. For writes, the W_HOLD field also defines the hold time for the data pins (EMIF_D). Refer to the datasheet of the external asynchronous device to determine the appropriate setting for this field.</p>
TA	<p><b>Minimum turnaround time.</b></p> <p>This field defines the minimum number of EMIF clock cycles between asynchronous reads and writes, minus one cycle. The purpose of this feature is to avoid contention on the bus. The value written to this field also determines the number of cycles that will be inserted between asynchronous accesses and SDRAM accesses. Refer to the datasheet of the external asynchronous device to determine the appropriate setting for this field.</p>

**Table 17-15. Description of the Asynchronous *m* Configuration Register (CE<sub>n</sub>CFG) (continued)**

Parameter	Description
ASIZE	<p><b>Asynchronous Device Bus Width.</b> This field determines the data bus width of the asynchronous interface in the following way:</p> <ul style="list-style-type: none"> <li>ASIZE = 0 selects an 8-bit bus</li> <li>ASIZE = 1 selects a 16-bit bus</li> </ul> <p>The configuration of ASIZE determines the function of the EMIF_A and EMIF_BA pins as described in <a href="#">Section 17.2.5.1</a>. This field also determines the number of external accesses required to fulfill a request generated by one of the sources mentioned in <a href="#">Section 17.2.2</a>. For example, a request for a 32-bit word would require four external access when ASIZE = 0. Refer to the datasheet of the external asynchronous device to determine the appropriate setting for this field.</p>

**Table 17-16. Description of the Asynchronous Wait Cycle Configuration Register (AWCC)**

Parameter	Description
WP <sub>n</sub>	<p><b>EM_WAIT Polarity.</b></p> <ul style="list-style-type: none"> <li>WP<sub>n</sub> = 0 selects active-low polarity</li> <li>WP<sub>n</sub> = 1 selects active-high polarity</li> </ul> <p>When set to 1, the EMIF will wait if the EMIF_nWAIT pin is high. When cleared to 0, the EMIF will wait if the EMIF_nWAIT pin is low. The EMIF must have the Extended Wait Mode enabled for the EMIF_nWAIT pin to affect the width of the strobe period.</p>
MAX_EXT_WAIT	<p><b>Maximum Extended Wait Cycles.</b> This field configures the number of EMIF clock cycles the EMIF will wait for the EMIF_nWAIT pin to be deactivated during the strobe period of an access cycle. The maximum number of EMIF clock cycles it will wait is determined by the following formula: Maximum Extended Wait Cycles = (MAX_EXT_WAIT + 1) × 16 If the EMIF_nWAIT pin is not deactivated within the time specified by this field, the EMIF resumes the access cycle, registering whatever data is on the bus and proceeding to the hold period of the access cycle. This situation is referred to as an Asynchronous Timeout. An Asynchronous Timeout generates an interrupt, if it has been enabled in the EMIF interrupt mask set register (INTMSKSET). Refer to <a href="#">Section 17.2.8.1</a> for more information about the EMIF interrupts.</p>

**Table 17-17. Description of the EMIF Interrupt Mask Set Register (INTMSKSET)**

Parameter	Description
WR_MASK_SET	<b>Wait Rise Mask Set.</b> Writing a 1 enables an interrupt to be generated when a rising edge on EMIF_nWAIT occurs while in NAND Flash Mode
AT_MASK_SET	<b>Asynchronous Timeout Mask Set.</b> Writing a 1 to this bit enables an interrupt to be generated when an Asynchronous Timeout occurs.

**Table 17-18. Description of the EMIF Interrupt Mast Clear Register (INTMSKCLR)**

Parameter	Description
WR_MASK_CLR	<b>Wait Rise Mask Clear.</b> Writing a 1 to this bit disables the interrupt, clearing the WR_MASK_SET bit in the EMIF interrupt mask set register (INTMSKSET).
AT_MASK_CLR	<b>Asynchronous Timeout Mask Clear.</b> Writing a 1 to this bit prevents an interrupt from being generated when an Asynchronous Timeout occurs.

#### 17.2.5.4 Read and Write Operations in Normal Mode

Normal Mode is the asynchronous interface's default mode of operation. It is selected when the SS bit in the asynchronous *n* configuration register (CE*n*CFG) is cleared to 0. In this mode, the EMIF\_nDQM pins operate as byte enables. [Section 17.2.5.4.1](#) and [Section 17.2.5.4.2](#) explain the details of read and write operations while in Normal Mode.

##### 17.2.5.4.1 Asynchronous Read Operations (Normal Mode)

---

**NOTE:** During an entire asynchronous read operation, the EMIF\_nWE pin is driven high.

---

An asynchronous read is performed when any of the requesters mentioned in [Section 17.2.2](#) request a read from the attached asynchronous memory. After the request is received, a read operation is initiated once it becomes the EMIF's highest priority task, according to the priority scheme detailed in [Section 17.2.12](#). In the event that the read request cannot be serviced by a single access cycle to the external device, multiple access cycles will be performed by the EMIF until the entire request is fulfilled. The details of an asynchronous read operation in Normal Mode are described in [Table 17-19](#). Also, [Figure 17-10](#) shows an example timing diagram of a basic read operation.

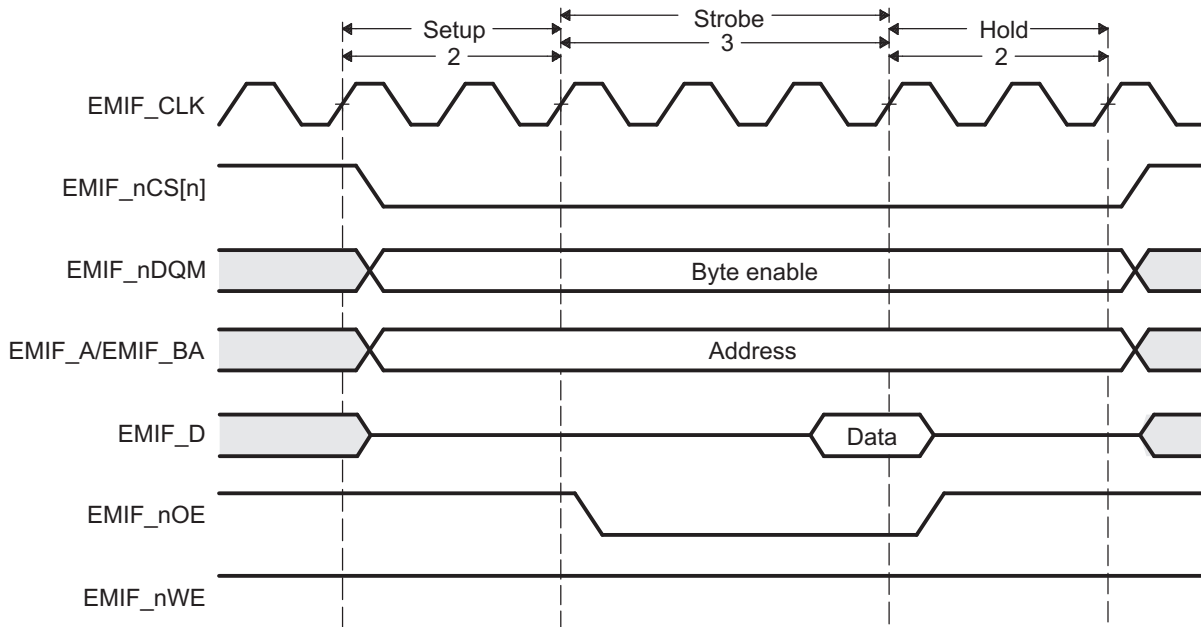
**Table 17-19. Asynchronous Read Operation in Normal Mode**

Time Interval	Pin Activity in Normal Mode
Turn-around period	<p>Once the read operation becomes the highest priority task for the EMIF, the EMIF waits for the programmed number of turn-around cycles before proceeding to the setup period of the operation. The number of wait cycles is taken directly from the TA field of the asynchronous <i>n</i> configuration register (CE<i>n</i>CFG). There are two exceptions to this rule:</p> <ul style="list-style-type: none"> <li>• If the current read operation was directly preceded by another read operation, no turnaround cycles are inserted.</li> <li>• If the current read operation was directly preceded by a write operation and the TA field has been cleared to 0, one turn-around cycle will be inserted.</li> </ul> <p>After the EMIF has waited for the turnaround cycles to complete, it again checks to make sure that the read operation is still its highest priority task. If so, the EMIF proceeds to the setup period of the operation. If it is no longer the highest priority task, the EMIF terminates the operation.</p>

**Table 17-19. Asynchronous Read Operation in Normal Mode (continued)**

Time Interval	Pin Activity in Normal Mode
Start of the setup period	<p>The following actions occur at the start of the setup period:</p> <ul style="list-style-type: none"> <li>The setup, strobe, and hold values are set according to the R_SETUP, R_STROBE, and R_HOLD values in CEnCFG.</li> <li>The address pins EMIF_A and EMIF_BA become valid and carry the values described in Section 17.2.5.1.</li> <li>EMIF_nCS[4:2] falls to enable the external device (if not already low from a previous operation)</li> </ul>
Strobe period	<p>The following actions occur during the strobe period of a read operation:</p> <ol style="list-style-type: none"> <li>EMIF_nOE falls at the start of the strobe period</li> <li>On the rising edge of the clock which is concurrent with the end of the strobe period: <ul style="list-style-type: none"> <li>EMIF_nOE rises</li> <li>The data on the EMIF_D bus is sampled by the EMIF.</li> </ul> </li> </ol> <p>In Figure 17-10, EMIF_nWAIT is inactive. If EMIF_nWAIT is instead activated, the strobe period can be extended by the external device to give it more time to provide the data. Section 17.2.5.6 contains more details on using the EMIF_nWAIT pin.</p>
End of the hold period	<p>At the end of the hold period:</p> <ul style="list-style-type: none"> <li>The address pins EMIF_A and EMIF_BA become invalid</li> <li>EMIF_nCS[4:2] rises (if no more operations are required to complete the current request)</li> </ul> <p>EMIF may be required to issue additional read operations to a device with a small data bus width in order to complete an entire word access. In this case, the EMIF immediately re-enters the setup period to begin another operation without incurring the turn-round cycle delay. The setup, strobe, and hold values are not updated in this case. If the entire word access has been completed, the EMIF returns to its previous state unless another asynchronous request has been submitted and is currently the highest priority task. If this is the case, the EMIF instead enters directly into the turnaround period for the pending read or write operation.</p>

**Figure 17-10. Timing Waveform of an Asynchronous Read Cycle in Normal Mode**





### 17.2.5.4.2 Asynchronous Write Operations (Normal Mode)

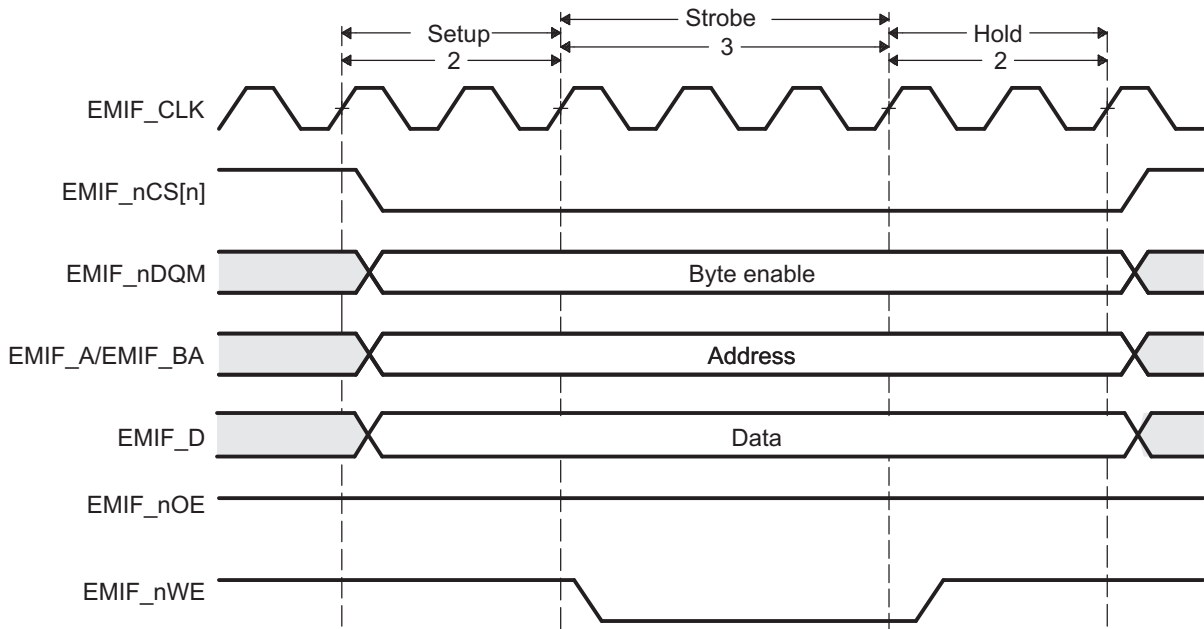
**NOTE:** During an entire asynchronous write operation, the EMIF\_nOE pin is driven high.

An asynchronous write is performed when any of the requesters mentioned in [Section 17.2.2](#) request a write to memory in the asynchronous bank of the EMIF. After the request is received, a write operation is initiated once it becomes the EMIF's highest priority task, according to the priority scheme detailed in [Section 17.2.12](#). In the event that the write request cannot be serviced by a single access cycle to the external device, multiple access cycles will be performed by the EMIF until the entire request is fulfilled. The details of an asynchronous write operation in Normal Mode are described in [Table 17-20](#). Also, [Figure 17-11](#) shows an example timing diagram of a basic write operation.

**Table 17-20. Asynchronous Write Operation in Normal Mode**

Time Interval	Pin Activity in Normal Mode
Turnaround period	<p>Once the write operation becomes the highest priority task for the EMIF, the EMIF waits for the programmed number of turn-around cycles before proceeding to the setup period of the operation. The number of wait cycles is taken directly from the TA field of the asynchronous <i>n</i> configuration register (CE<sub>n</sub>CFG). There are two exceptions to this rule:</p> <ul style="list-style-type: none"> <li>• If the current write operation was directly preceded by another write operation, no turn-around cycles are inserted.</li> <li>• If the current write operation was directly preceded by a read operation and the TA field has been cleared to 0, one turnaround cycle will be inserted.</li> </ul> <p>After the EMIF has waited for the turn-around cycles to complete, it again checks to make sure that the write operation is still its highest priority task. If so, the EMIF proceeds to the setup period of the operation. If it is no longer the highest priority task, the EMIF terminates the operation.</p>
Start of the setup period	<p>The following actions occur at the start of the setup period:</p> <ul style="list-style-type: none"> <li>• The setup, strobe, and hold values are set according to the W_SETUP, W_STROBE, and W_HOLD values in CE<sub>n</sub>CFG.</li> <li>• The address pins EMIF_A and EMIF_BA and the data pins EMIF_D become valid. The EMIF_A and EMIF_BA pins carry the values described in <a href="#">Section 17.2.5.1</a>.</li> <li>• EMIF_nCS[4:2] falls to enable the external device (if not already low from a previous operation).</li> </ul>
Strobe period	<p>The following actions occur at the start of the strobe period of a write operation:</p> <ol style="list-style-type: none"> <li>1. EMIF_nWE falls</li> <li>2. The EMIF_nDQM pins become valid as byte enables.</li> </ol> <p>The following actions occur on the rising edge of the clock which is concurrent with the end of the strobe period:</p> <ol style="list-style-type: none"> <li>1. EMIF_nWE rises</li> <li>2. The EMIF_nDQM pins deactivate</li> </ol> <p>In <a href="#">Figure 17-11</a>, EMIF_nWAIT is inactive. If EMIF_nWAIT is instead activated, the strobe period can be extended by the external device to give it more time to accept the data. <a href="#">Section 17.2.5.6</a> contains more details on using the EMIF_nWAIT pin.</p>
End of the hold period	<p>At the end of the hold period:</p> <ul style="list-style-type: none"> <li>• The address pins EMIF_A and EMIF_BA become invalid</li> <li>• The data pins become invalid</li> <li>• EMIF_nCS[n] (n = 2, 3, or 4) rises (if no more operations are required to complete the current request)</li> </ul> <p>The EMIF may be required to issue additional write operations to a device with a small data bus width in order to complete an entire word access. In this case, the EMIF immediately re-enters the setup period to begin another operation without incurring the turnaround cycle delay. The setup, strobe, and hold values are not updated in this case. If the entire word access has been completed, the EMIF returns to its previous state unless another asynchronous request has been submitted and is currently the highest priority task. If this is the case, the EMIF instead enters directly into the turnaround period for the pending read or write operation.</p>

**Figure 17-11. Timing Waveform of an Asynchronous Write Cycle in Normal Mode**



### 17.2.5.5 Read and Write Operation in Select Strobe Mode

Select Strobe Mode is the EMIF's second mode of operation. It is selected when the SS bit of the asynchronous  $n$  configuration register (CE $n$ CFG) is set to 1. In this mode, the EMIF\_nDQM pins operate as byte enables and the EMIF\_nCS[n] ( $n = 2, 3, \text{ or } 4$ ) pin is only active during the strobe period of an access cycle. [Section 17.2.5.4.1](#) and [Section 17.2.5.4.2](#) explain the details of read and write operations while in Select Strobe Mode.

#### 17.2.5.5.1 Asynchronous Read Operations (Select Strobe Mode)

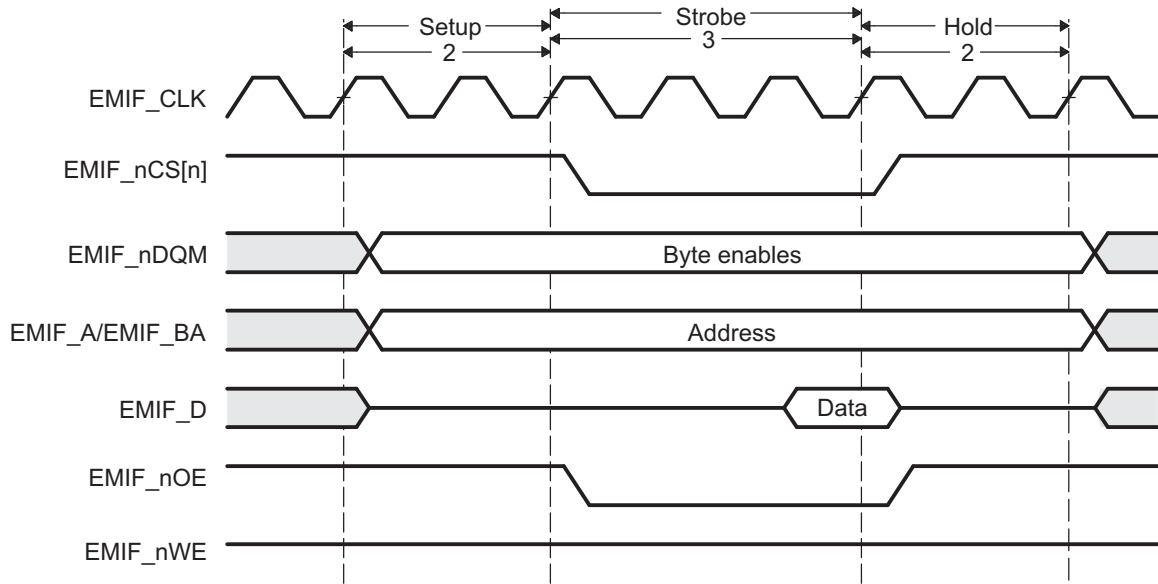
**NOTE:** During the entirety of an asynchronous read operation, the EMIF\_nWE pin is driven high.

An asynchronous read is performed when any of the requesters mentioned in [Section 17.2.2](#) request a read from the attached asynchronous memory. After the request is received, a read operation is initiated once it becomes the EMIF's highest priority task, according to the priority scheme detailed in [Section 17.2.12](#). In the event that the read request cannot be serviced by a single access cycle to the external device, multiple access cycles will be performed by the EMIF until the entire request is fulfilled. The details of an asynchronous read operation in Select Strobe Mode are described in [Table 17-21](#). Also, [Figure 17-12](#) shows an example timing diagram of a basic read operation.

**Table 17-21. Asynchronous Read Operation in Select Strobe Mode**

Time Interval	Pin Activity in Select Strobe Mode
Turnaround period	<p>Once the read operation becomes the highest priority task for the EMIF, the EMIF waits for the programmed number of turn-around cycles before proceeding to the setup period of the operation. The number of wait cycles is taken directly from the TA field of the asynchronous <math>n</math> configuration register (CE<math>n</math>CFG). There are two exceptions to this rule:</p> <ul style="list-style-type: none"> <li>• If the current read operation was directly preceded by another read operation, no turn-around cycles are inserted.</li> <li>• If the current read operation was directly preceded by a write operation and the TA field has been cleared to 0, one turn-around cycle will be inserted.</li> </ul> <p>After the EMIF has waited for the turn-around cycles to complete, it again checks to make sure that the read operation is still its highest priority task. If so, the EMIF proceeds to the setup period of the operation. If it is no longer the highest priority task, the EMIF terminates the operation.</p>
Start of the setup period	<p>The following actions occur at the start of the setup period:</p> <ul style="list-style-type: none"> <li>• The setup, strobe, and hold values are set according to the R_SETUP, R_STROBE, and R_HOLD values in CE<math>n</math>CFG.</li> <li>• The address pins EMIF_A and EMIF_BA become valid and carry the values described in <a href="#">Section 17.2.5.1</a>.</li> <li>• The EMIF_nDQM pins become valid as byte enables.</li> </ul>
Strobe period	<p>The following actions occur during the strobe period of a read operation:</p> <ol style="list-style-type: none"> <li>1. EMIF_nCS[n] (<math>n = 2, 3, \text{ or } 4</math>) and EMIF_nOE fall at the start of the strobe period</li> <li>2. On the rising edge of the clock which is concurrent with the end of the strobe period: <ul style="list-style-type: none"> <li>• EMIF_nCS[n] (<math>n = 2, 3, \text{ or } 4</math>) and EMIF_nOE rise</li> <li>• The data on the EMIF_D bus is sampled by the EMIF.</li> </ul> </li> </ol> <p>In <a href="#">Figure 17-12</a>, EMIF_nWAIT is inactive. If EMIF_nWAIT is instead activated, the strobe period can be extended by the external device to give it more time to provide the data. <a href="#">Section 17.2.5.6</a> contains more details on using the EMIF_nWAIT pin.</p>
End of the hold period	<p>At the end of the hold period:</p> <ul style="list-style-type: none"> <li>• The address pins EMIF_A and EMIF_BA become invalid</li> <li>• The EMIF_nDQM pins become invalid</li> </ul> <p>The EMIF may be required to issue additional read operations to a device with a small data bus width in order to complete an entire word access. In this case, the EMIF immediately re-enters the setup period to begin another operation without incurring the turnaround cycle delay. The setup, strobe, and hold values are not updated in this case. If the entire word access has been completed, the EMIF returns to its previous state unless another asynchronous request has been submitted and is currently the highest priority task. If this is the case, the EMIF instead enters directly into the turnaround period for the pending read or write operation.</p>

**Figure 17-12. Timing Waveform of an Asynchronous Read Cycle in Select Strobe Mode**



### 17.2.5.5.2 Asynchronous Write Operations (Select Strobe Mode)

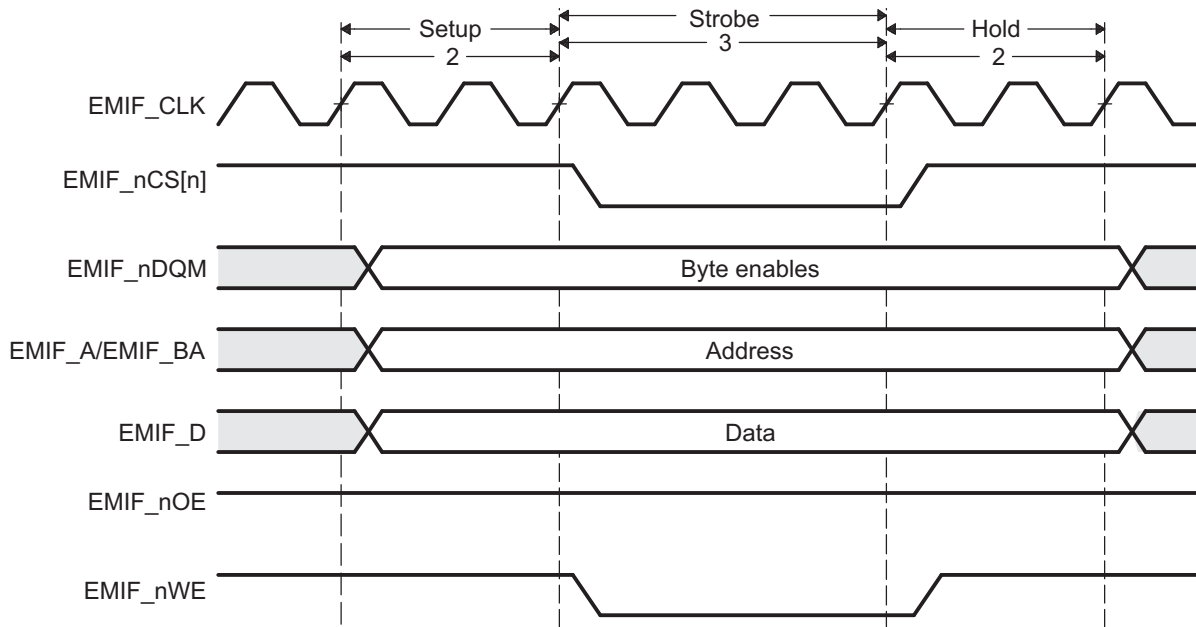
**NOTE:** During the entirety of an asynchronous write operation, the EMIF\_nOE pin is driven high.

An asynchronous write is performed when any of the requesters mentioned in [Section 17.2.2](#) request a write to memory in the asynchronous bank of the EMIF. After the request is received, a write operation is initiated once it becomes the EMIF's highest priority task, according to the priority scheme detailed in [Section 17.2.12](#). In the event that the write request cannot be serviced by a single access cycle to the external device, multiple access cycles will be performed by the EMIF until the entire request is fulfilled. The details of an asynchronous write operation in Select Strobe Mode are described in [Table 17-22](#). Also, [Figure 17-13](#) shows an example timing diagram of a basic write operation.

**Table 17-22. Asynchronous Write Operation in Select Strobe Mode**

Time Interval	Pin Activity in Select Strobe Mode
Turnaround period	<p>Once the write operation becomes the highest priority task for the EMIF, the EMIF waits for the programmed number of turnaround cycles before proceeding to the setup period of the operation. The number of wait cycles is taken directly from the TA field of the asynchronous <i>n</i> configuration register (CE<sub>n</sub>CFG). There are two exceptions to this rule:</p> <ul style="list-style-type: none"> <li>• If the current write operation was directly preceded by another write operation, no turn-around cycles are inserted.</li> <li>• If the current write operation was directly preceded by a read operation and the TA field has been cleared to 0, one turnaround cycle will be inserted.</li> </ul> <p>After the EMIF has waited for the turnaround cycles to complete, it again checks to make sure that the write operation is still its highest priority task. If so, the EMIF proceeds to the setup period of the operation. If it is no longer the highest priority task, the EMIF terminates the operation.</p>
Start of the setup period	<p>The following actions occur at the start of the setup period:</p> <ul style="list-style-type: none"> <li>• The setup, strobe, and hold values are set according to the W_SETUP, W_STROBE, and W_HOLD values in CE<sub>n</sub>CFG.</li> <li>• The address pins EMIF_A and EMIF_BA and the data pins EMIF_D become valid. The EMIF_A and EMIF_BA pins carry the values described in <a href="#">Section 17.2.5.1</a>.</li> <li>• The EMIF_nDQM pins become active as byte enables.</li> </ul>
Strobe period	<p>The following actions occur at the start of the strobe period of a write operation:</p> <ul style="list-style-type: none"> <li>• EMIF_nCS[n] (n = 2, 3, or 4) and EMIF_nWE fall</li> </ul> <p>The following actions occur on the rising edge of the clock which is concurrent with the end of the strobe period:</p> <ul style="list-style-type: none"> <li>• EMIF_nCS[n] (n = 2, 3, or 4) and EMIF_nWE rise</li> </ul> <p>In <a href="#">Figure 17-13</a>, EMIF_nWAIT is inactive. If EMIF_nWAIT is instead activated, the strobe period can be extended by the external device to give it more time to accept the data. <a href="#">Section 17.2.5.6</a> contains more details on using the EMIF_nWAIT pin.</p>
End of the hold period	<p>At the end of the hold period:</p> <ul style="list-style-type: none"> <li>• The address pins EMIF_A and EMIF_BA become invalid</li> <li>• The data pins become invalid</li> <li>• The EMIF_nDQM pins become invalid</li> </ul> <p>The EMIF may be required to issue additional write operations to a device with a small data bus width in order to complete an entire word access. In this case, the EMIF immediately re-enters the setup period to begin another operation without incurring the turnaround cycle delay. The setup, strobe, and hold values are not updated in this case. If the entire word access has been completed, the EMIF returns to its previous state unless another asynchronous request has been submitted and is currently the highest priority task. If this is the case, the EMIF instead enters directly into the turn-around period for the pending read or write operation.</p>

**Figure 17-13. Timing Waveform of an Asynchronous Write Cycle in Select Strobe Mode**



#### 17.2.5.6 Extended Wait Mode and the EMIF\_nWAIT Pin

The EMIF supports the Extend Wait Mode. This is a mode in which the external asynchronous device may assert control over the length of the strobe period. The Extended Wait Mode can be entered by setting the EW bit in the asynchronous  $n$  configuration register (CENCFG) ( $n = 2, 3, \text{ or } 4$ ). When this bit is set, the EMIF monitors the EMIF\_nWAIT pin to determine if the attached device wishes to extend the strobe period of the current access cycle beyond the programmed number of clock cycles.

When the EMIF detects that the EMIF\_nWAIT pin has been asserted, it will begin inserting extra strobe cycles into the operation until the EMIF\_nWAIT pin is deactivated by the external device. The EMIF will then return to the last cycle of the programmed strobe period and the operation will proceed as usual from this point. Please refer to the device data manual for details on the timing requirements of the EMIF\_nWAIT signal.

The EMIF\_nWAIT pin cannot be used to extend the strobe period indefinitely. The programmable MAX\_EXT\_WAIT field in the asynchronous wait cycle configuration register (AWCC) determines the maximum number of EMIF\_CLK cycles the strobe period may be extended beyond the programmed length. When the counter expires, the EMIF proceeds to the hold period of the operation regardless of the state of the EMIF\_nWAIT pin. The EMIF can also generate an interrupt upon expiration of this counter. See Section 17.2.8.1 for details on enabling this interrupt.

For the EMIF to function properly in the Extended Wait mode, the WP $n$  bit of AWCC must be programmed to match the polarity of the EMIF\_nWAIT pin. In its reset state of 1, the EMIF will insert wait cycles when the EMIF\_nWAIT pin is sampled high. When set to 0, the EMIF will insert wait cycles only when EMIF\_nWAIT is sampled low. This programmability allows for a glueless connection to larger variety of asynchronous devices.

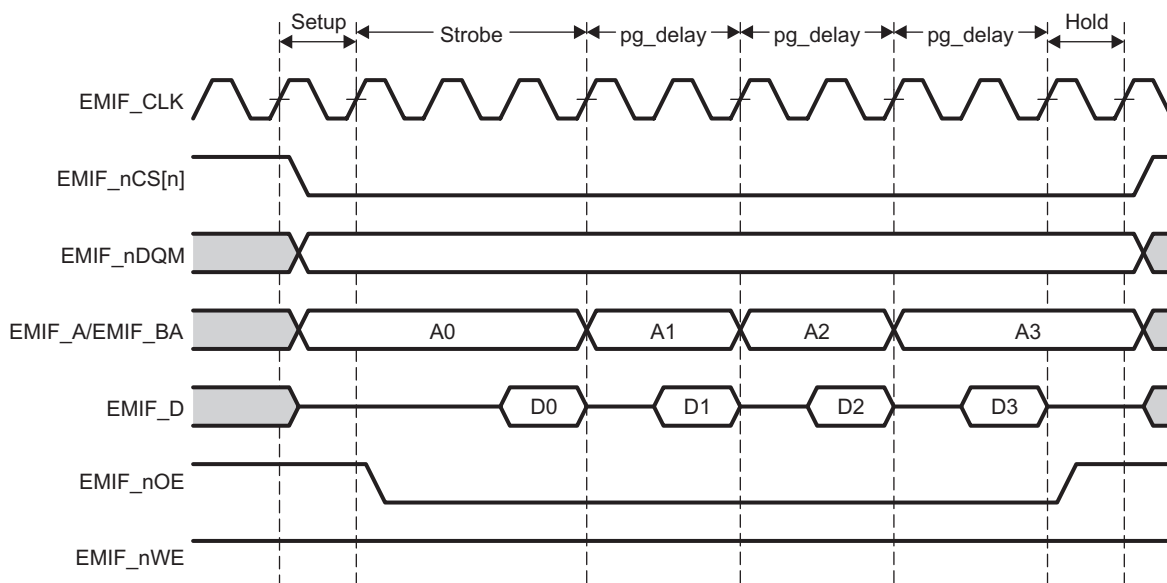
Finally, a restriction is placed on the strobe period timing parameters when operating in Extended Wait mode. Specifically, the sum of the W\_SETUP and W\_STROBE fields must be greater than 4, and the sum of the R\_SETUP and R\_STROBE fields must be greater than 4 for the EMIF to recognize the EMIF\_nWAIT pin has been asserted. The W\_SETUP, W\_STROBE, R\_SETUP, and R\_STROBE fields are in CENCFG.

### 17.2.5.7 NOR Flash Page Mode

EMIF supports Page mode reads for NOR Flash on its asynchronous memory chip selects. This mode can be enabled by writing a 1 to the  $CS_n\_PG\_MD\_EN$  ( $n = 2, 3, \text{ or } 4$ ) field in the Page Mode Control register for the chip select in consideration. Whenever Page Mode for reads is enabled for a particular chip select, the page size for the device connected must also be programmed in the  $CS_n\_PG\_SIZE$  field of the Page Mode Control register. The address change to valid read data available timing must be programmed in the  $CS_n\_PG\_DEL$  field of the Page Control register. All other asynchronous memory timings must be programmed in the asynchronous configuration register (CE $n$ CFG). See [Figure 17-14](#) for read in asynchronous page mode.

**NOTE:** The Extended Wait mode and the Select Strobe mode must be disabled when using the asynchronous interface in Page mode.

**Figure 17-14. Asynchronous Read in Page Mode**



### 17.2.6 Data Bus Parking

The EMIF always drives the data bus to the previous write data value when it is idle. This feature is called data bus parking. Only when the EMIF issues a read command to the external memory does it stop driving the data bus. After the EMIF latches the last read data, it immediately parks the data bus again.

The one exception to this behavior occurs after performing an asynchronous read operation while the EMIF is in the self-refresh state. In this situation, the read operation is not followed by the EMIF parking the data bus. Instead, the EMIF tri-states the data bus. Therefore, it is not recommended to perform asynchronous read operations while the EMIF is in the self-refresh state, in order to prevent floating inputs on the data bus. External pull-ups, such as 10k $\Omega$  resistors, should be placed on the 16 EMIF data bus pins (which do not have internal pull-ups) if it is required to perform reads in this situation. The precise resistor value should be chosen so that the worst case combined off-state leakage currents do not cause the voltage levels on the associated pins to drop below the high-level input voltage requirement.

For information about the self-refresh state, see [Section 17.2.4.7](#).

### 17.2.7 Reset and Initialization Considerations

The EMIF memory controller has two active-low reset signals, CHIP\_RST\_n and MOD\_G\_RST\_n. Both these reset signals are driven by the device system reset signal. This device does not offer the flexibility to reset just the EMIF state machine without also resetting the EMIF controller's memory-mapped registers. As soon as the device system reset is released (driven High), the EMIF memory controller immediately begins its initialization sequence. Command and data stored in the EMIF memory controller FIFOs are lost. Refer the Architecture chapter of the technical reference manual (TRM) for more information on conditions that can cause a device system reset to be asserted.

When system reset is released, the EMIF automatically begins running the SDRAM initialization sequence described in [Section 17.2.4.4](#). Even though the initialization procedure is automatic, a special procedure, found in [Section 17.2.4.5](#) must still be followed.

### 17.2.8 Interrupt Support

The EMIF supports a single interrupt to the CPU. [Section 17.2.8.1](#) details the generation and internal masking of EMIF interrupts.

#### 17.2.8.1 Interrupt Events

There are three conditions that may cause the EMIF to generate an interrupt to the CPU. These conditions are:

- A rising edge on the EMIF\_nWAIT signal (wait rise interrupt)
- An asynchronous time out
- Usage of unsupported addressing mode (line trap interrupt)

The wait rise interrupt occurs when a rising edge is detected on EMIF\_nWAIT signal. This interrupt generation is not affected by the WP<sub>n</sub> bit in the asynchronous wait cycle configuration register (AWCC). The asynchronous time out interrupt condition occurs when the attached asynchronous device fails to deassert the EMIF\_nWAIT pin within the number of cycles defined by the MAX\_EXT\_WAIT bit in AWCC (this happens only in extended wait mode). EMIF supports only linear incrementing and cache line wrap addressing modes. If an access request for an unsupported addressing mode is received, the EMIF will set the LT bit in the EMIF interrupt raw register (INTRAW) and treat the request as a linear incrementing request.

Only when the interrupt is enabled by setting the appropriate bit (WR\_MASK\_SET/AT\_MASK\_SET/LT\_MASK\_SET) in the EMIF interrupt mask set register (INTMSKSET) to 1, will the interrupt be sent to the CPU. Once enabled, the interrupt may be disabled by writing a 1 to the corresponding bit in the EMIF interrupt mask clear register (INTMSKCLR). The bit fields in both the INTMSKSET and INTMSKCLR may be used to indicate whether the interrupt is enabled. When the interrupt is enabled, the corresponding bit field in both the INTMSKSET and INTMSKCLR will have a value of 1; when the interrupt is disabled, the corresponding bit field will have a value of 0.

The EMIF interrupt raw register (INTRAW) and the EMIF interrupt mask register (INTMSK) indicate the status of each interrupt. The appropriate bit (WR/AT/LT) in INTRAW is set when the interrupt condition occurs, whether or not the interrupt has been enabled. However, the appropriate bit (WR\_MASKED/AT\_MASKED/LT\_MASKED) in INTMSK is set only when the interrupt condition occurs and the interrupt is enabled. Writing a 1 to the bit in INTRAW clears the INTRAW bit as well as the corresponding bit in INTMSK. [Table 17-23](#) contains a brief summary of the interrupt status and control bit fields. See [Section 17.3](#) for complete details on the register fields.

**Table 17-23. Interrupt Monitor and Control Bit Fields**

Register Name	Bit Name	Description
EMIF interrupt raw register (INTRAW)	WR	This bit is set when an rising edge on the EMIF_nWAIT signal occurs. Writing a 1 clears the WR bit as well as the WR_MASKED bit in INTMSK.
	AT	This bit is set when an asynchronous timeout occurs. Writing a 1 clears the AT bit as well as the AT_MASKED bit in INTMSK.
	LT	This bit is set when an unsupported addressing mode is used. Writing a 1 clears LT bit as well as the LT_MASKED bit in INTMSK.



**Table 17-23. Interrupt Monitor and Control Bit Fields (continued)**

Register Name	Bit Name	Description
EMIF interrupt mask register (INTMSK)	WR_MASKED	This bit is set only when a rising edge on the EMIF_nWAIT signal occurs and the interrupt has been enabled by writing a 1 to the WR_MASK_SET bit in INTMSKSET.
	AT_MASKED	This bit is set only when an asynchronous timeout occurs and the interrupt has been enabled by writing a 1 to the AT_MASK_SET bit in INTMSKSET.
	LT_MASKED	This bit is set only when line trap interrupt occurs and the interrupt has been enabled by writing a 1 to the LT_MASK_SET bit in INTMSKSET.
EMIF interrupt mask set register (INTMSKSET)	WR_MASK_SET	Writing a 1 to this bit enables the wait rise interrupt.
	AT_MASK_SET	Writing a 1 to this bit enables the asynchronous timeout interrupt.
	LT_MASK_SET	Writing a 1 to this bit enables the line trap interrupt.
EMIF interrupt mask clear register (INTMSKCLR)	WR_MASK_CLR	Writing a 1 to this bit disables the wait rise interrupt.
	AT_MASK_CLR	Writing a 1 to this bit disables the asynchronous timeout interrupt.
	LT_MASK_CLR	Writing a 1 to this bit disables the line trap interrupt.

### 17.2.9 DMA Event Support

EMIF memory controller is a DMA slave peripheral and therefore does not generate DMA events. Data read and write requests may be made directly, by masters and the DMA.

### 17.2.10 EMIF Signal Multiplexing

For details on EMIF signal multiplexing, see the I/O Multiplexing Module chapter of the technical reference manual.

### 17.2.11 Memory Map

For information describing the device memory-map, see your device-specific datasheet.

### 17.2.12 Priority and Arbitration

[Section 17.2.2](#) describes the external prioritization and arbitration among requests from different sources within the microcontroller. The result of this external arbitration is that only one request is presented to the EMIF at a time. Once the EMIF completes a request, the external arbiter then provides the EMIF with the next pending request.

Internally, the EMIF undertakes memory device transactions according to a strict priority scheme. The highest priority events are:

- A device reset.
- A write to any of the three least significant bytes of the SDRAM configuration register (SDCR).

Either of these events will cause the EMIF to immediately commence its initialization sequence as described in [Section 17.2.4.4](#).

Once the EMIF has completed its initialization sequence, it performs memory transactions according to the following priority scheme (highest priority listed first):

1. If the EMIF's backlog refresh counter is at the Refresh Must urgency level, the EMIF performs multiple SDRAM auto refresh cycles until the Refresh Release urgency level is reached.
2. If an SDRAM or asynchronous read has been requested, the EMIF performs a read operation.
3. If the EMIF's backlog refresh counter is at the Refresh Need urgency level, the EMIF performs an SDRAM auto refresh cycle.
4. If an SDRAM or asynchronous write has been requested, the EMIF performs a write operation.
5. If the EMIF's backlog refresh counter is at the Refresh May or Refresh Release urgency level, the EMIF performs an SDRAM auto refresh cycle.

6. If the value of the SR bit in SDCR has been set to 1, the EMIF will enter the self-refresh state as described in [Section 17.2.4.7](#).

After taking one of the actions listed above, the EMIF then returns to the top of the priority list to determine its next action.

Because the EMIF does not issue auto-refresh cycles when in the self-refresh state, the above priority scheme does not apply when in this state. See [Section 17.2.4.7](#) for details on the operation of the EMIF when in the self-refresh state.

## 17.2.13 System Considerations

This section describes various system considerations to keep in mind when operating the EMIF.

### 17.2.13.1 Asynchronous Request Times

In a system that interfaces to both SDRAM and asynchronous memory, the asynchronous requests must not take longer than the smaller of the following two values:

- $t_{\text{RAS}}$  (typically 120  $\mu\text{s}$ ) - to avoid violating the maximum time allowed between issuing an ACTV and PRE command to the SDRAM.
- $t_{\text{Refresh Rate}} \times 11$  (typically 15.7  $\mu\text{s} \times 11 = 172.7 \mu\text{s}$ ) - to avoid refresh violations on the SDRAM.

The length of an asynchronous request is controlled by multiple factors, the primary factor being the number of access cycles required to complete the request. For example, an asynchronous request for 4 bytes will require four access cycles using an 8-bit data bus and only two access cycle using a 16-bit data bus. The maximum request size that the EMIF can be sent is 16 words, therefore the maximum number of access cycles per memory request is 64 when the EMIF is configured with an 8-bit data bus. The length of the individual access cycles that make up the asynchronous request is determined by the programmed setup, strobe, hold, and turnaround values, but can also be extended with the assertion of the EMIF\_nWAIT input signal up to a programmed maximum limit. It is up to the user to make sure that an entire asynchronous request does not exceed the timing values listed above when also interfacing to an SDRAM device. This can be done by limiting the asynchronous timing parameters.

### **17.2.14 Power Management**

Power dissipation from the EMIF memory controller may be managed by following methods:

- Self-refresh mode
- Power-down mode
- Gating input clocks to the module off

Gating input clocks off to the EMIF memory controller achieves higher power savings when compared to the power savings of self-refresh or power down mode. The input clock VCLK3 can be turned off through the use of the Global Clock Module (GCM). Before gating clocks off, the EMIF memory controller must place the SDR SDRAM memory in self-refresh mode. If the external memory requires a continuous clock, the VCLK3 clock domain must not be turned off because this may result in data corruption. See the following subsections for the proper procedures to follow when stopping the EMIF memory controller clocks.

#### **17.2.14.1 Power Management Using Self-Refresh Mode**

The EMIF can be placed into a self-refresh state in order to place the attached SDRAM devices into self-refresh mode, which consumes less power for most SDRAM devices. In this state, the attached SDRAM device uses an internal clock to perform its own auto refresh cycles. This maintains the validity of the data in the SDRAM without the need for any external commands. Refer to [Section 17.2.4.7](#) for more details on placing the EMIF into the self-refresh state.

#### **17.2.14.2 Power Management Using Power Down Mode**

In the power down mode, EMIF drives EMIF\_CKE low to lower the power consumption. EMIF\_CKE goes high when there is a need to send refresh (REFR) commands, after which EMIF\_CKE is again driven low. EMIF\_CKE remains low until any request arrives. Refer to [Section 17.2.4.8](#) for more details on placing EMIF in power down mode.

### **17.2.15 Emulation Considerations**

EMIF memory controller remains fully functional during emulation halts in order to allow emulation access to external memory.

## 17.3 Registers

The external memory interface (EMIF) is controlled by programming its internal memory-mapped registers (MMRs). [Table 17-24](#) lists the memory-mapped registers for the EMIF.

---

**NOTE:** All EMIF MMRs, except SDCR, support only word (32-bit) accesses. Performing a byte (8-bit) or halfword (16-bit) write to these registers results in undefined behavior. The SDCR is byte writable to allow the setting of the SR, PD and PDWR bits without triggering the SDRAM initialization sequence.

---

The EMIF registers must always be accessed using 32-bit accesses (unless otherwise specified in this chapter). The base address of the memory-mapped registers of EMIF is 0xFCFFE800.

**Table 17-24. External Memory Interface (EMIF) Registers**

Offset	Acronym	Register Description	Section
0h	MIDR	Module ID Register	<a href="#">Section 17.3.1</a>
4h	AWCC	Asynchronous Wait Cycle Configuration Register	<a href="#">Section 17.3.2</a>
8h	SDCR	SDRAM Configuration Register	<a href="#">Section 17.3.3</a>
Ch	SDRCR	SDRAM Refresh Control Register	<a href="#">Section 17.3.4</a>
10h	CE2CFG	Asynchronous 1 Configuration Register	<a href="#">Section 17.3.5</a>
14h	CE3CFG	Asynchronous 2 Configuration Register	<a href="#">Section 17.3.5</a>
18h	CE4CFG	Asynchronous 3 Configuration Register	<a href="#">Section 17.3.5</a>
1Ch	CE5CFG	Asynchronous 4 Configuration Register	<a href="#">Section 17.3.5</a>
20h	SDTIMR	SDRAM Timing Register	<a href="#">Section 17.3.6</a>
3Ch	SDSRETR	SDRAM Self Refresh Exit Timing Register	<a href="#">Section 17.3.7</a>
40h	INTRAW	EMIF Interrupt Raw Register	<a href="#">Section 17.3.8</a>
44h	INTMSK	EMIF Interrupt Mask Register	<a href="#">Section 17.3.9</a>
48h	INTMSKSET	EMIF Interrupt Mask Set Register	<a href="#">Section 17.3.10</a>
4Ch	INTMSKCLR	EMIF Interrupt Mask Clear Register	<a href="#">Section 17.3.11</a>
68h	PMCR	Page Mode Control Register	<a href="#">Section 17.3.12</a>

### 17.3.1 Module ID Register (MIDR)

This is a read-only register indicating the module ID of the EMIF. The MIDR is shown in [Figure 17-15](#) and described in [Table 17-25](#).

**Figure 17-15. Module ID Register (MIDR)**



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

**Table 17-25. Module ID Register (MIDR) Field Descriptions**

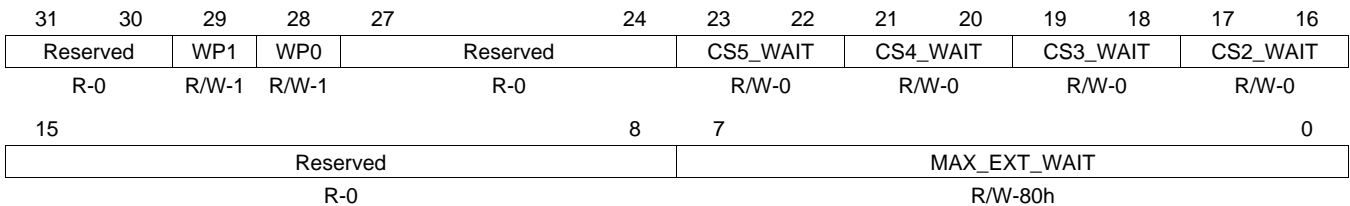
Bit	Field	Value	Description
31-0	REV	4000 0205h	Module ID of EMIF.

### 17.3.2 Asynchronous Wait Cycle Configuration Register (AWCC)

The asynchronous wait cycle configuration register (AWCC) is used to configure the parameters for extended wait cycles. Both the polarity of the EMIF\_nWAIT pin(s) and the maximum allowable number of extended wait cycles can be configured. The AWCC is shown in [Figure 17-16](#) and described in [Table 17-26](#). Not all devices support both EMIF\_nWAIT[1] and EMIF\_nWAIT[0], see the device-specific data manual to determine support on each device.

**NOTE:** The EW bit in the asynchronous *n* configuration register (CE<sub>n</sub>CFG) must be set to allow for the insertion of extended wait cycles.

**Figure 17-16. Asynchronous Wait Cycle Configuration Register (AWCCR)**



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

**Table 17-26. Asynchronous Wait Cycle Configuration Register (AWCCR) Field Descriptions**

Bit	Field	Value	Description
31-30	Reserved	0	Reserved
29	WP1	0 1	EMIF_nWAIT[1] polarity bit. This bit defines the polarity of the EMIF_nWAIT[1] pin. Insert wait cycles if EMIF_nWAIT[1] pin is low. Insert wait cycles if EMIF_nWAIT[1] pin is high.
28	WP0	0 1	EMIF_nWAIT[0] polarity bit. This bit defines the polarity of the EMIF_nWAIT[0] pin. Insert wait cycles if EMIF_nWAIT[0] pin is low. Insert wait cycles if EMIF_nWAIT[0] pin is high.
27-24	Reserved	0	Reserved
23-22	CS5_WAIT	0-3h	Chip Select 5 WAIT signal selection. This signal determines which EMIF_nWAIT[n] signal will be used for memory accesses to chip select 5 memory space. This device does not support chip select 5, so any value written to this field has no effect.
21-20	CS4_WAIT	0-3h 0 1h 2h-3h	Chip Select 4 WAIT signal selection. This signal determines which EMIF_nWAIT[n] signal will be used for memory accesses to chip select 4 memory space. EMIF_nWAIT[0] pin is used to control external wait states. EMIF_nWAIT[1] pin is used to control external wait states. Reserved
19-18	CS3_WAIT	0-3h 0 1h 2h-3h	Chip Select 3 WAIT signal selection. This signal determines which EMIF_nWAIT[n] signal will be used for memory accesses to chip select 3 memory space. EMIF_nWAIT[0] pin is used to control external wait states. EMIF_nWAIT[1] pin is used to control external wait states. Reserved
17-16	CS2_WAIT	0-3h 0 1h 2h-3h	Chip Select 2 WAIT signal selection. This signal determines which EMIF_nWAIT[n] signal will be used for memory accesses to chip select 2 memory space. EMIF_nWAIT[0] pin is used to control external wait states.. EMIF_nWAIT[1] pin is used to control external wait states. Reserved
15-8	Reserved	0	Reserved
7-0	MAX_EXT_WAIT	0-FFh	Maximum extended wait cycles. The EMIF will wait for a maximum of (MAX_EXT_WAIT + 1) × 16 clock cycles before it stops inserting asynchronous wait cycles and proceeds to the hold period of the access.

### 17.3.3 SDRAM Configuration Register (SDCR)

The SDRAM configuration register (SDCR) is used to configure various parameters of the SDRAM controller such as the number of internal banks, the internal page size, and the CAS latency to match those of the attached SDRAM device. In addition, this register is used to put the attached SDRAM device into Self-Refresh mode. The SDCR is shown in Figure 17-17 and described in Table 17-27.

**NOTE:** Writing to the lower three bytes of this register will cause the EMIF to start the SDRAM initialization sequence described in Section 17.2.4.4.

**Figure 17-17. SDRAM Configuration Register (SDCR)**

31	30	29	28	24		
SR	PD	PDWR	Reserved			
R/W-0	R/W-0	R/W-0	R-0			
23	Reserved			16		
R-0						
15	14	13	12	11	9	8
Reserved	NM <sup>(A)</sup>	Reserved		CL		BIT11_9LOCK
R-0	R/W-0	R-0		R/W-3h		R/W-0
7	6	4		3	2	0
Reserved	IBANK		Reserved	PAGESIZE		
R-0	R/W-2h		R-0	R/W-0		

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

A. The NM bit must be set to 1 if the EMIF on your device only has 16 data bus pins.

**Table 17-27. SDRAM Configuration Register (SDCR) Field Descriptions**

Bit	Field	Value	Description
31	SR	0 1	Self-Refresh mode bit. This bit controls entering and exiting of the Self-Refresh mode described in Section 17.2.4.7. The field should be written using a byte-write to the upper byte of SDCR to avoid triggering the SDRAM initialization sequence. Writing a 0 to this bit will cause connected SDRAM devices and the EMIF to exit the Self-Refresh mode. Writing a 1 to this bit will cause connected SDRAM devices and the EMIF to enter the Self-Refresh mode.
30	PD	0 1	Power Down bit. This bit controls entering and exiting of the power-down mode. The field should be written using a byte-write to the upper byte of SDCR to avoid triggering the SDRAM initialization sequence. If both SR and PD bits are set, the EMIF will go into Self Refresh. Writing a 0 to this bit will cause connected SDRAM devices and the EMIF to exit the power-down mode. Writing a 1 to this bit will cause connected SDRAM devices and the EMIF to enter the power-down mode.
29	PDWR		Perform refreshes during power down. Writing a 1 to this bit will cause EMIF to exit power-down state and issue and AUTO REFRESH command every time Refresh May level is set.
28-15	Reserved	0	Reserved. The reserved bit location is always read as 0. If writing to this field, always write the default value of 0.
14	NM	0 1	Narrow mode bit. This bit defines whether a 16- or 32-bit-wide SDRAM is connected to the EMIF. This bit field must always be set to 1. Writing to this field triggers the SDRAM initialization sequence. 0 32-bit SDRAM data bus is used. 1 16-bit SDRAM data bus is used.
13-12	Reserved	0	Reserved. The reserved bit location is always read as 0. If writing to this field, always write the default value of 0.



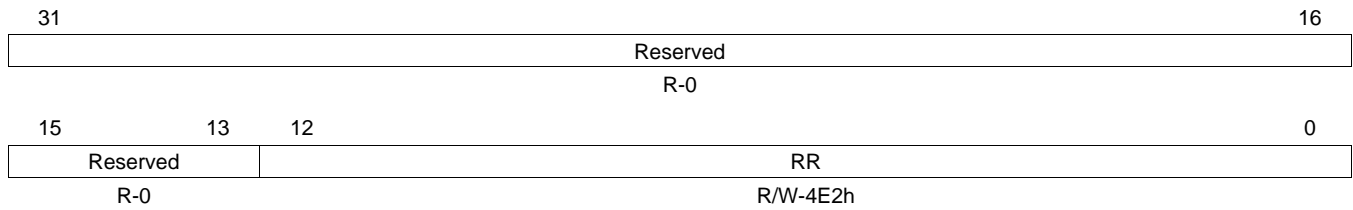
**Table 17-27. SDRAM Configuration Register (SDCR) Field Descriptions (continued)**

Bit	Field	Value	Description
11-9	CL	0-7h 0-1h 2h 3h 4h-7h	CAS Latency. This field defines the CAS latency to be used when accessing connected SDRAM devices. A 1 must be simultaneously written to the BIT11_9LOCK bit field of this register in order to write to the CL bit field. Writing to this field triggers the SDRAM initialization sequence. Reserved CAS latency = 2 EMIF_CLK cycles CAS latency = 3 EMIF_CLK cycles Reserved
8	BIT11_9LOCK	0 1	Bits 11 to 9 lock. CL can only be written if BIT11_9LOCK is simultaneously written with a 1. BIT11_9LOCK is always read as 0. Writing to this field triggers the SDRAM initialization sequence. CL cannot be written. CL can be written.
7	Reserved	0	Reserved. The reserved bit location is always read as 0. If writing to this field, always write the default value of 0.
6-4	IBANK	0-7h 0 1 2 3h-7h	Internal SDRAM Bank size. This field defines number of banks inside the connected SDRAM devices. Writing to this field triggers the SDRAM initialization sequence. 1 bank SDRAM devices. 2 bank SDRAM devices. 4 bank SDRAM devices. Reserved.
3	Reserved	0	Reserved. The reserved bit location is always read as 0. If writing to this field, always write the default value of 0.
2-0	PAGESIZE	0-7h 0 1h 2h 3h 4h-7h	Page Size. This field defines the internal page size of connected SDRAM devices. Writing to this field triggers the SDRAM initialization sequence. 8 column address bits (256 elements per row) 9 column address bits (512 elements per row) 10 column address bits (1024 elements per row) 11 column address bits (2048 elements per row) Reserved

### 17.3.4 SDRAM Refresh Control Register (SDRCR)

The SDRAM refresh control register (SDRCR) is used to configure the rate at which connected SDRAM devices will be automatically refreshed by the EMIF. Refer to [Section 17.2.4.6](#) on the refresh controller for more details. The SDRCR is shown in [Figure 17-18](#) and described in [Table 17-28](#).

**Figure 17-18. SDRAM Refresh Control Register (SDRCR)**



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

**Table 17-28. SDRAM Refresh Control Register (SDRCR) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reserved. The reserved bit location is always read as 0. If writing to this field, always write the default value of 0.
12-0	RR	0-1FFFh	Refresh Rate. This field is used to define the SDRAM refresh period in terms of EMIF_CLK cycles. Writing a value < 0x0020 to this field will cause it to be loaded with (2 × T_RFC) + 1 value from the SDRAM timing register (SDTIMR).

### 17.3.5 Asynchronous *n* Configuration Registers (CE2CFG-CE5CFG)

The asynchronous *n* configuration registers (CE2CFG, CE3CFG, CE4CFG, and CE5CFG) are used to configure the shaping of the address and control signals during an access to asynchronous memory connected to CS2, CS3, CS4, and CS5, respectively. CS5 is not available on this device. It is also used to program the width of asynchronous Interfacend to select from various modes of operation. This register can be written prior to any transfer, and any asynchronous transfer following the write will use the new configuration. The CE $n$ CFG is shown in Figure 17-19 and described in Table 17-29.

**Figure 17-19. Asynchronous *n* Configuration Register (CE $n$ CFG)**

31	30	29	26	25	24
SS	EW <sup>(A)</sup>	W_SETUP		W_STROBE <sup>(B)</sup>	
R/W-0	R/W-0	R/W-Fh		R/W-3Fh	
23	20		19	17	16
W_STROBE <sup>(B)</sup>			W_HOLD		R_SETUP
R/W-3Fh			R/W-7h		R/W-Fh
15	13	12	7	6	4
R_SETUP		R_STROBE <sup>(B)</sup>		R_HOLD	TA
R/W-Fh		R/W-3Fh		R/W-7h	R/W-3h
				3	2
				1	0
				ASIZE	
				R/W-0	

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

A. The EW bit must be cleared to 0 when operating in NAND Flash mode.

B. This bit field must be cleared to 0 if the EMIF on your device does not have an EMIF\_nWAIT pin.

**Table 17-29. Asynchronous *n* Configuration Register (CE $n$ CFG) Field Descriptions**

Bit	Field	Value	Description
31	SS	0 1	Select Strobe bit. This bit defines whether the asynchronous interface operates in Normal Mode or Select Strobe Mode. See Section 17.2.5 for details on the two modes of operation. Normal Mode enabled. Select Strobe Mode enabled.
30	EW	0 1	Extend Wait bit. This bit defines whether extended wait cycles will be enabled. See Section 17.2.5.6 on extended wait cycles for details. This bit field must be set to 0 if the EMIF on your device does not have an EMIF_nWAIT pin. Extended wait cycles disabled. Extended wait cycles enabled.
29-26	W_SETUP	0-Fh	Write setup width in EMIF_CLK cycles, minus one cycle. See Section 17.2.5.3 for details.
25-20	W_STROBE	0-3Fh	Write strobe width in EMIF_CLK cycles, minus one cycle. See Section 17.2.5.3 for details.
19-17	W_HOLD	0-7h	Write hold width in EMIF_CLK cycles, minus one cycle. See Section 17.2.5.3 for details.
16-13	R_SETUP	0-Fh	Read setup width in EMIF_CLK cycles, minus one cycle. See Section 17.2.5.3 for details.
12-7	R_STROBE	0-3Fh	Read strobe width in EMIF_CLK cycles, minus one cycle. See Section 17.2.5.3 for details.
6-4	R_HOLD	0-7h	Read hold width in EMIF_CLK cycles, minus one cycle. See Section 17.2.5.3 for details.
3-2	TA	0-3h	Minimum Turn-Around time. This field defines the minimum number of EMIF_CLK cycles between reads and writes, minus one cycle. See Section 17.2.5.3 for details.
1-0	ASIZE	0-3h 0 1h 2h-3h	Asynchronous Data Bus Width. This field defines the width of the asynchronous device's data bus. 8-bit data bus 16-bit data bus Reserved

### 17.3.6 SDRAM Timing Register (SDTIMR)

The SDRAM timing register (SDTIMR) is used to program many of the SDRAM timing parameters. Consult the SDRAM datasheet for information on the appropriate values to program into each field. The SDTIMR is shown in [Figure 17-20](#) and described in [Table 17-30](#).

**Figure 17-20. SDRAM Timing Register (SDTIMR)**

31	27	26	24	23	22	20	19	18	16	
T_RFC			T_RP		Rsvd	T_RCD		Rsvd	T_WR	
R/W-8h			R/W-2h		R-0	R/W-2h		R-0	R/W-1h	
15	12	11	8	7	6	4	3			
T_RAS		T_RC			Rsvd	T_RRD		Reserved		
R/W-5h		R/W-8h			R-0	R/W-1h		R-0		

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

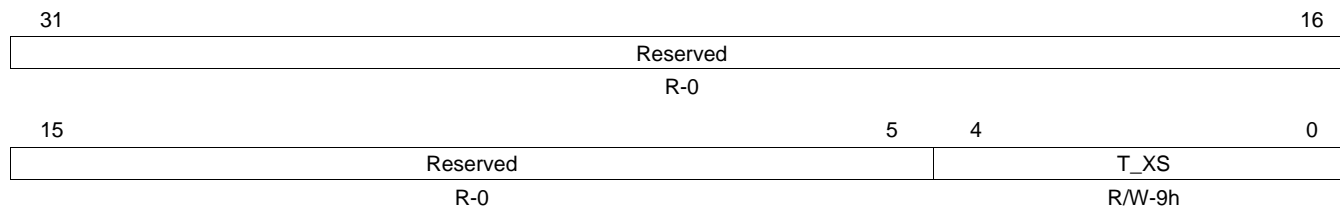
**Table 17-30. SDRAM Timing Register (SDTIMR) Field Descriptions**

Bit	Field	Value	Description
31-27	T_RFC	0-1Fh	Specifies the Trfc value of the SDRAM. This defines the minimum number of EMIF_CLK cycles from Refresh (REFR) to Refresh (REFR), minus 1: $T\_RFC = (Trfc/t_{EMIF\_CLK}) - 1$
26-24	T_RP	0-7h	Specifies the Trp value of the SDRAM. This defines the minimum number of EMIF_CLK cycles from Precharge (PRE) to Activate (ACTV) or Refresh (REFR) command, minus 1: $T\_RP = (Trp/t_{EMIF\_CLK}) - 1$
23	Reserved	0	Reserved. The reserved bit location is always read as 0. If writing to this field, always write the default value of 0.
22-20	T_RCD	0-7h	Specifies the Trcd value of the SDRAM. This defines the minimum number of EMIF_CLK cycles from Active (ACTV) to Read (READ) or Write (WRT), minus 1: $T\_RCD = (Trcd/t_{EMIF\_CLK}) - 1$
19	Reserved	0	Reserved. The reserved bit location is always read as 0. If writing to this field, always write the default value of 0.
18-16	T_WR	0-7h	Specifies the Twr value of the SDRAM. This defines the minimum number of EMIF_CLK cycles from last Write (WRT) to Precharge (PRE), minus 1: $T\_WR = (Twr/t_{EMIF\_CLK}) - 1$
15-12	T_RAS	0-Fh	Specifies the Tras value of the SDRAM. This defines the minimum number of EMIF_CLK clock cycles from Activate (ACTV) to Precharge (PRE), minus 1: $T\_RAS = (Tras/t_{EMIF\_CLK}) - 1$
11-8	T_RC	0-Fh	Specifies the Trc value of the SDRAM. This defines the minimum number of EMIF_CLK clock cycles from Activate (ACTV) to Activate (ACTV), minus 1: $T\_RC = (Trc/t_{EMIF\_CLK}) - 1$
7	Reserved	0	Reserved. The reserved bit location is always read as 0. If writing to this field, always write the default value of 0.
6-4	T_RRD	0-7h	Specifies the Trrd value of the SDRAM. This defines the minimum number of EMIF_CLK clock cycles from Activate (ACTV) to Activate (ACTV) for a different bank, minus 1: $T\_RRD = (Trrd/t_{EMIF\_CLK}) - 1$
3-0	Reserved	0	Reserved. The reserved bit location is always read as 0. If writing to this field, always write the default value of 0.

### 17.3.7 SDRAM Self Refresh Exit Timing Register (SDSRETR)

The SDRAM self refresh exit timing register (SDSRETR) is used to program the amount of time between when the SDRAM exits Self-Refresh mode and when the EMIF issues another command. The SDSRETR is shown in [Figure 17-21](#) and described in [Table 17-31](#).

**Figure 17-21. SDRAM Self Refresh Exit Timing Register (SDSRETR)**



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

**Table 17-31. SDRAM Self Refresh Exit Timing Register (SDSRETR) Field Descriptions**

Bit	Field	Value	Description
31-5	Reserved	0	Reserved. The reserved bit location is always read as 0.
4-0	T_XS	0-1Fh	This field specifies the minimum number of ECLKOUT cycles from Self-Refresh exit to any command, minus one. $T\_XS = T_{xsr} / t_{EMIF\_CLK} - 1$

### 17.3.8 EMIF Interrupt Raw Register (INTRAW)

The EMIF interrupt raw register (INTRAW) is used to monitor and clear the EMIF's hardware-generated Asynchronous Timeout Interrupt. The AT bit in this register will be set when an Asynchronous Timeout occurs regardless of the status of the EMIF interrupt mask set register (INTMSKSET) and EMIF interrupt mask clear register (INTMSKCLR). Writing a 1 to this bit will clear it. The EMIF on some devices does not have the EMIF\_nWAIT pin; therefore, these registers and fields are reserved on those devices. The INTRAW is shown in [Figure 17-22](#) and described in [Table 17-32](#).

**Figure 17-22. EMIF Interrupt Raw Register (INTRAW)**

31	Reserved				8
R-0					
7	3	2	1	0	
Reserved		WR	LT	AT	
R-0		R/W1C-0	R/W1C-0	R/W1C-0	

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear (writing 0 has no effect); -n = value after reset

**Table 17-32. EMIF Interrupt Raw Register (INTRAW) Field Descriptions**

Bit	Field	Value	Description
31-3	Reserved	0	Reserved. The reserved bit location is always read as 0. If writing to this field, always write the default value of 0.
2	WR	0 1	Wait Rise. This bit is set to 1 by hardware to indicate that a rising edge on the EMIF_nWAIT pin has occurred. 0 Indicates that a rising edge has not occurred on the EMIF_nWAIT pin. Writing a 0 has no effect. 1 Indicates that a rising edge has occurred on the EMIF_nWAIT pin. Writing a 1 will clear this bit and the WR_MASKED bit in the EMIF interrupt masked register (INTMSK).
1	LT	0 1	Line Trap. Set to 1 by hardware to indicate illegal memory access type or invalid cache line size. 0 Writing a 0 has no effect. 1 Indicates that a line trap has occurred. Writing a 1 will clear this bit as well as the LT_MASKED bit in the EMIF interrupt masked register (INTMSK).
0	AT	0 1	Asynchronous Timeout. This bit is set to 1 by hardware to indicate that during an extended asynchronous memory access cycle, the EMIF_nWAIT pin did not go inactive within the number of cycles defined by the MAX_EXT_WAIT field in the asynchronous wait cycle configuration register (AWCC). 0 Indicates that an Asynchronous Timeout has not occurred. Writing a 0 has no effect. 1 Indicates that an Asynchronous Timeout has occurred. Writing a 1 will clear this bit as well as the AT_MASKED bit in the EMIF interrupt masked register (INTMSK).

### 17.3.9 EMIF Interrupt Masked Register (INTMSK)

Like the EMIF interrupt raw register (INTRAW), the EMIF interrupt masked register (INTMSK) is used to monitor and clear the status of the EMIF's hardware-generated Asynchronous Timeout Interrupt. The main difference between the two registers is that when the AT\_MASKED bit in this register is set, an active-high pulse will be sent to the CPU interrupt controller. Also, the AT\_MASKED bit field in INTMSK is only set to 1 if the associated interrupt has been enabled in the EMIF interrupt mask set register (INTMSKSET). The EMIF on some devices does not have the EMIF\_nWAIT pin, therefore, these registers and fields are reserved on those devices. The INTMSK is shown in Figure 17-23 and described in Table 17-33.

**Figure 17-23. EMIF Interrupt Mask Register (INTMSK)**

31	Reserved				8
R-0					
7	3	2	1	0	
Reserved		WR_MASKED	LT_MASKED	AT_MASKED	
R-0		R/W1C-0	R/W1C-0	R/W1C-0	

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear (writing 0 has no effect); -n = value after reset

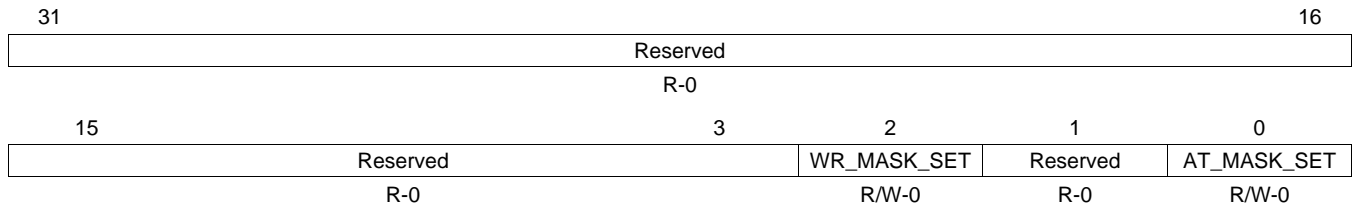
**Table 17-33. EMIF Interrupt Mask Register (INTMSK) Field Descriptions**

Bit	Field	Value	Description
31-3	Reserved	0	Reserved. The reserved bit location is always read as 0. If writing to this field, always write the default value of 0.
2	WR_MASKED	0	Wait Rise Masked. This bit is set to 1 by hardware to indicate a rising edge has occurred on the EMIF_nWAIT pin, provided that the WR_MASK_SET bit is set to 1 in the EMIF interrupt mask set register (INTMSKSET).
		1	Indicates that a wait rise interrupt has not been generated. Writing a 0 has no effect.
		1	Indicates that a wait rise interrupt has been generated. Writing a 1 will clear this bit and the WR bit in the EMIF interrupt raw register (INTRAW).
1	LT_MASKED	0	Masked Line Trap. Set to 1 by hardware to indicate illegal memory access type or invalid cache line size, only if the LT_MASK_SET bit in the EMIF interrupt mask set register (INTMSKSET) is set to 1.
		1	Writing a 0 has no effect.
		1	Writing a 1 will clear this bit as well as the LT bit in the EMIF interrupt raw register (INTRAW).
0	AT_MASKED	0	Asynchronous Timeout Masked. This bit is set to 1 by hardware to indicate that during an extended asynchronous memory access cycle, the EMIF_nWAIT pin did not go inactive within the number of cycles defined by the MAX_EXT_WAIT field in the asynchronous wait cycle configuration register (AWCC), provided that the AT_MASK_SET bit is set to 1 in the EMIF interrupt mask set register (INTMSKSET).
		1	Indicates that an Asynchronous Timeout Interrupt has not been generated. Writing a 0 has no effect.
		1	Indicates that an Asynchronous Timeout Interrupt has been generated. Writing a 1 will clear this bit as well as the AT bit in the EMIF interrupt raw register (INTRAW).

### 17.3.10 EMIF Interrupt Mask Set Register (INTMSKSET)

The EMIF interrupt mask set register (INTMSKSET) is used to enable the Asynchronous Timeout Interrupt. If read as 1, the AT\_MASKED bit in the EMIF interrupt masked register (INTMSK) will be set and an interrupt will be generated when an Asynchronous Timeout occurs. If read as 0, the AT\_MASKED bit will always read 0 and no interrupt will be generated when an Asynchronous Timeout occurs. Writing a 1 to the AT\_MASK\_SET bit enables the Asynchronous Timeout Interrupt. The EMIF on some devices does not have the EMIF\_nWAIT pin; therefore, these registers and fields are reserved on those devices. The INTMSKSET is shown in [Figure 17-24](#) and described in [Table 17-34](#).

**Figure 17-24. EMIF Interrupt Mask Set Register (INTMSKSET)**



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

**Table 17-34. EMIF Interrupt Mask Set Register (INTMSKSET) Field Descriptions**

Bit	Field	Value	Description
31-3	Reserved	0	Reserved. The reserved bit location is always read as 0. If writing to this field, always write the default value of 0.
2	WR_MASK_SET	0	Indicates that the wait rise interrupt is disabled. Writing a 0 has no effect.
		1	Indicates that the wait rise interrupt is enabled. Writing a 1 sets this bit and the WR_MASK_CLR bit in the EMIF interrupt mask clear register (INTMSKCLR).
1	LT_MASK_SET	0	Indicates that the line trap interrupt is disabled. Writing a 0 has no effect.
		1	Indicates that the line trap interrupt is enabled. Writing a 1 sets this bit and the LT_MASK_CLR bit in the EMIF interrupt mask clear register (INTMSKCLR).
0	AT_MASK_SET	0	Indicates that the Asynchronous Timeout Interrupt is disabled. Writing a 0 has no effect.
		1	Indicates that the Asynchronous Timeout Interrupt is enabled. Writing a 1 sets this bit and the AT_MASK_CLR bit in the EMIF interrupt mask clear register (INTMSKCLR).



### 17.3.11 EMIF Interrupt Mask Clear Register (INTMSKCLR)

The EMIF interrupt mask clear register (INTMSKCLR) is used to disable the Asynchronous Timeout Interrupt. If read as 1, the AT\_MASKED bit in the EMIF interrupt masked register (INTMSK) will be set and an interrupt will be generated when an Asynchronous Timeout occurs. If read as 0, the AT\_MASKED bit will always read 0 and no interrupt will be generated when an Asynchronous Timeout occurs. Writing a 1 to the AT\_MASK\_CLR bit disables the Asynchronous Timeout Interrupt. The EMIF on some devices does not have the EMIF\_nWAIT pin, therefore, these registers and fields are reserved on those devices. The INTMSKCLR is shown in [Figure 17-25](#) and described in [Table 17-35](#).

**Figure 17-25. EMIF Interrupt Mask Clear Register (INTMSKCLR)**

31	Reserved				16
R-0					
15	3	2	1	0	
Reserved		WR_MASK_CLR	Reserved	AT_MASK_CLR	
R-0		R/W-0	R-0	R/W-0	

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

**Table 17-35. EMIF Interrupt Mask Clear Register (INTMSKCLR) Field Descriptions**

Bit	Field	Value	Description
31-3	Reserved	0	Reserved. The reserved bit location is always read as 0. If writing to this field, always write the default value of 0.
2	WR_MASK_CLR	0	Indicates that the wait rise interrupt is disabled. Writing a 0 has no effect.
		1	Indicates that the wait rise interrupt is enabled. Writing a 1 clears this bit and the WR_MASK_SET bit in the EMIF interrupt mask set register (INTMSKSET).
1	LT_MASK_CLR	0	Indicates that the line trap interrupt is disabled. Writing a 0 has no effect.
		1	Indicates that the line trap interrupt is enabled. Writing a 1 clears this bit and the LT_MASK_SET bit in the EMIF interrupt mask set register (INTMSKSET).
0	AT_MASK_CLR	0	Indicates that the Asynchronous Timeout Interrupt is disabled. Writing a 0 has no effect.
		1	Indicates that the Asynchronous Timeout Interrupt is enabled. Writing a 1 clears this bit and the AT_MASK_SET bit in the EMIF interrupt mask set register (INTMSKSET).

### 17.3.12 Page Mode Control Register (PMCR)

The page mode control register (PMCR) is shown in [Figure 17-26](#) and described in [Table 17-36](#). This register is configured when using NOR Flash page mode.

**Figure 17-26. Page Mode Control Register (PMCR)**

31	26	25	24
CS5_PG_DEL		CS5_PG_SIZE	CS5_PG_MD_EN
R/W-3Fh		R/W-0	R/W-0
23	18	17	16
CS4_PG_DEL		CS4_PG_SIZE	CS4_PG_MD_EN
R/W-3Fh		R/W-0	R/W-0
15	10	9	8
CS3_PG_DEL		CS3_PG_SIZE	CS3_PG_MD_EN
R/W-3Fh		R/W-0	R/W-0
7	2	1	0
CS2_PG_DEL		CS2_PG_SIZE	CS2_PG_MD_EN
R/W-3Fh		R/W-0	R/W-0

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

**Table 17-36. Page Mode Control Register (PMCR) Field Descriptions**

Bit	Field	Value	Description
31-26	CS5_PG_DEL	1-3Fh	Page access delay for NOR Flash connected on CS5. CS5 is not available on this device.
25	CS5_PG_SIZE		Page Size for NOR Flash connected on CS5. CS5 is not available on this device.
24	CS5_PG_MD_EN		Page Mode enable for NOR Flash connected on CS5. CS5 is not available on this device.
23-18	CS4_PG_DEL	1-3Fh	Page access delay for NOR Flash connected on CS4. Number of EMIF_CLK cycles required for the page read data to be valid, minus one cycle. This value must not be cleared to 0.
17	CS4_PG_SIZE		Page Size for NOR Flash connected on CS4.
		0	Page size is 4 words
		1	Page size is 8 words
16	CS4_PG_MD_EN		Page Mode enable for NOR Flash connected on CS4.
		0	Page mode disabled for this chip select
		1	Page mode enabled for this chip select
15-10	CS3_PG_DEL	1-3Fh	Page access delay for NOR Flash connected on CS3. Number of EMIF_CLK cycles required for the page read data to be valid, minus one cycle. This value must not be cleared to 0.
9	CS3_PG_SIZE		Page Size for NOR Flash connected on CS3.
		0	Page size is 4 words
		1	Page size is 8 words
8	CS3_PG_MD_EN		Page Mode enable for NOR Flash connected on CS3.
		0	Page mode disabled for this chip select
		1	Page mode enabled for this chip select
7-2	CS2_PG_DEL	1-3Fh	Page access delay for NOR Flash connected on CS2. Number of EMIF_CLK cycles required for the page read data to be valid, minus one cycle. This value must not be cleared to 0.
1	CS2_PG_SIZE		Page Size for NOR Flash connected on CS2.
		0	Page size is 4 words
		1	Page size is 8 words
0	CS2_PG_MD_EN		Page Mode enable for NOR Flash connected on CS2.
		0	Page mode disabled for this chip select
		1	Page mode enabled for this chip select

## 17.4 Example Configuration

This section presents an example of interfacing the EMIF to both an SDR SDRAM device and an asynchronous flash device.

### 17.4.1 Hardware Interface

Figure 17-27 shows the hardware interface between the EMIF, a Samsung K4S641632H-TC(L)70 64Mb SDRAM device, and two SHARP LH28F800BJE-PTTL90 8Mb Flash memory. The connection between the EMIF and the SDRAM is straightforward, but the connection between the EMIF and the flash deserves a detailed look.

The address inputs for the flash are provided by three sources. The A[18:0] address inputs are provided by a combination of the EMIF\_A and EMIF\_BA pins according to Section 17.2.5.1. RD/nBY signal from one flash is connected to EMIF\_nWAIT pin of EMIF.

Finally, this example configuration connects the EMIF\_nWE pin to the nWE input of the flash and operates the EMIF in Select Strobe Mode.

### 17.4.2 Software Configuration

The following sections describe how to configure the EMIF registers and bit fields to interface the EMIF with the Samsung K4S641632H-TC(L)70 SDRAM and the SHARP LH28F800BJE-PTTL90 8Mb Flash memory.

#### 17.4.2.1 Configuring the SDRAM Interface

This section describes how to configure the EMIF to interface with the Samsung K4S641632H-TC(L)70 SDRAM with a clock frequency of  $f_{EMIF\_CLK} = 100$  MHz. Procedure A described in Section 17.2.4.5 is followed which assumes that the SDRAM power-up timing constraint were met during the SDRAM Auto-Initialization sequence after Reset.

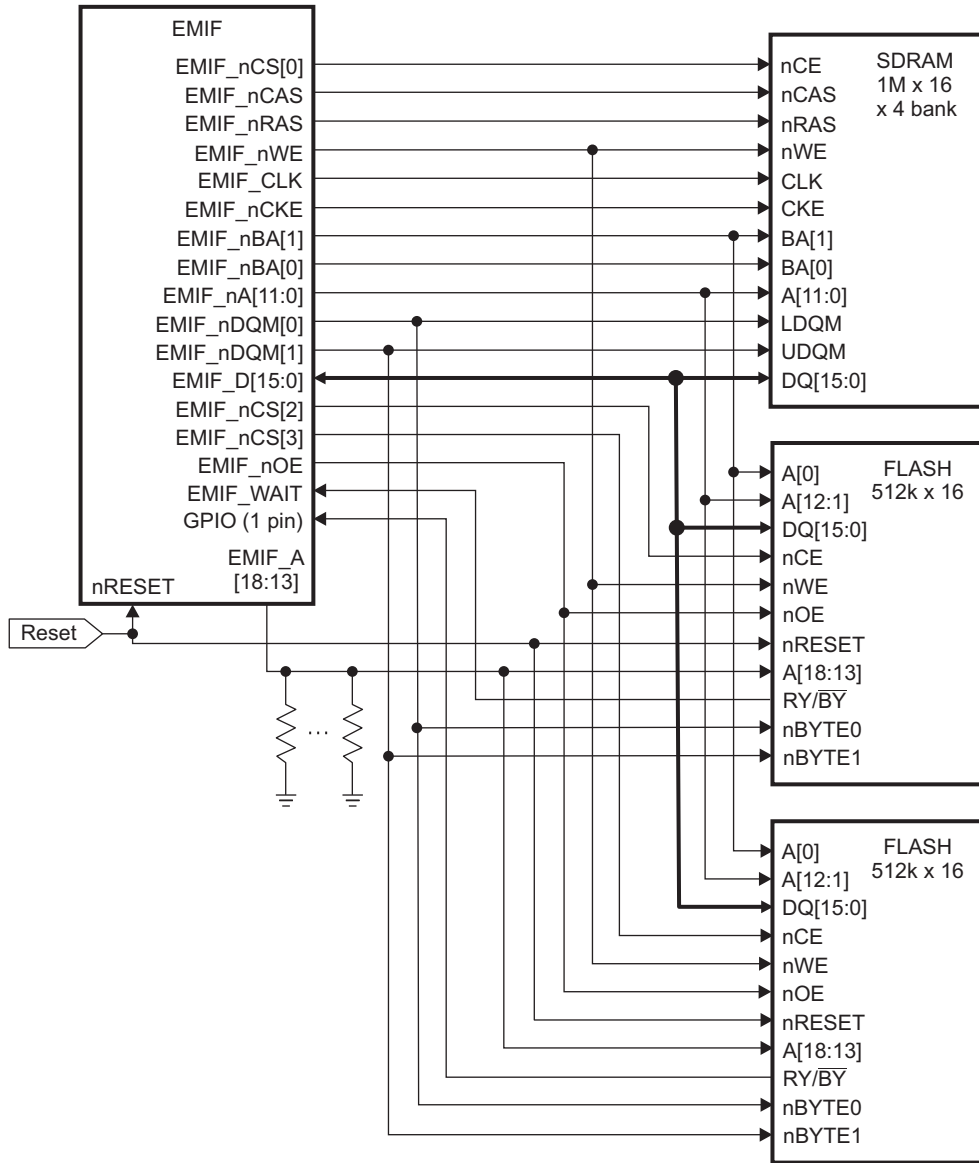
##### 17.4.2.1.1 PLL Programming for the EMIF to K4S641632H-TC(L)70 Interface

The device global clock module (GCM) should first be programmed to select the desired EMIF\_CLK frequency. Before doing this, the SDRAM should be placed in Self-Refresh Mode by setting the SR bit in the SDRAM configuration register (SDCR). The SR bit should be set using a byte-write to the upper byte of the SDCR to avoid triggering the SDRAM Initialization Sequence. The EMIF\_CLK frequency can now be configured to the desired value by selecting the appropriate clock source for the VCLK3 domain. Once the VCLK3 domain frequency has been configured, remove the SDRAM from Self-Refresh by clearing the SR bit in SDCR, again with a byte-write.

**Table 17-37. SR Field Value For the EMIF to K4S641632H-TC(L)70 Interface**

Field	Value	Purpose
SR	1 then 0	To place the EMIF into the self refresh state

Figure 17-27. Example Configuration Interface



### 17.4.2.1.2 SDRAM Timing Register (SDTIMR) Settings for the EMIF to K4S641632H-TC(L)70 Interface

The fields of the SDRAM timing register (SDTIMR) should be programmed first as described in [Table 17-38](#) to satisfy the required timing parameters for the K4S641632H-TC(L)70. Based on these calculations, a value of 6111 4610h should be written to SDTIMR. [Figure 17-28](#) shows a graphical description of how SDTIMR should be programmed.

**Table 17-38. SDTIMR Field Calculations for the EMIF to K4S641632H-TC(L)70 Interface**

Field Name	Formula	Value from K4S641632H-TC(L)70 Datasheet	Value Calculated for Field
T_RFC	$T\_RFC \geq (t_{RFC} \times f_{EMIF\_CLK}) - 1$	$t_{RC} = 68 \text{ ns (min)}^{(1)}$	6
T_RP	$T\_RP \geq (t_{RP} \times f_{EMIF\_CLK}) - 1$	$t_{RP} = 20 \text{ ns (min)}$	1
T_RCD	$T\_RCD \geq (t_{RCD} \times f_{EMIF\_CLK}) - 1$	$t_{RCD} = 20 \text{ ns (min)}$	1
T_WR	$T\_WR \geq (t_{WR} \times f_{EMIF\_CLK}) - 1$	$t_{RDL} = 2 \text{ CLK} = 20 \text{ ns (min)}^{(2)}$	1
T_RAS	$T\_RAS \geq (t_{RAS} \times f_{EMIF\_CLK}) - 1$	$t_{RAS} = 49 \text{ ns (min)}$	4
T_RC	$T\_RC \geq (t_{RC} \times f_{EMIF\_CLK}) - 1$	$t_{RC} = 68 \text{ ns (min)}$	6
T_RRD	$T\_RRD \geq (t_{RRD} \times f_{EMIF\_CLK}) - 1$	$t_{RRD} = 14 \text{ ns (min)}$	1

<sup>(1)</sup> The Samsung datasheet does not specify a  $t_{RFC}$  value. Instead, Samsung specifies  $t_{RC}$  as the minimum auto refresh period.

<sup>(2)</sup> The Samsung datasheet does not specify a  $t_{WR}$  value. Instead, Samsung specifies  $t_{RDL}$  as last data in to row precharge minimum delay.

**Figure 17-28. SDRAM Timing Register (SDTIMR)**

31	27	26	24	23	22	20	19	18	16	
0 0110			001		0	001		0	001	
T_RFC			T_RP		Rsvd	T_RCD		Rsvd	T_WR	
15	12	11	8	7	6	4	3	0		
0100		0110			0	001		0000		
T_RAS		T_RC			Rsvd	T_RRD		Reserved		

**17.4.2.1.3 SDRAM Self Refresh Exit Timing Register (SDSRETR) Settings for the EMIF to K4S641632H-TC(L)70 Interface**

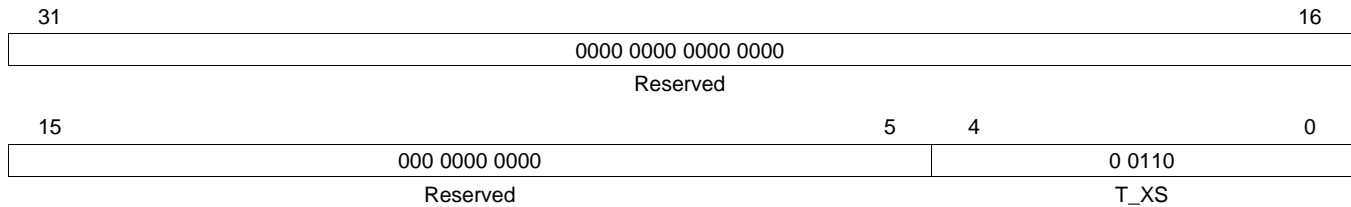
The SDRAM self refresh exit timing register (SDSRETR) should be programmed second to satisfy the  $t_{XSR}$  timing requirement from the K4S641632H-TC(L)70 datasheet. Table 17-39 shows the calculation of the proper value to program into the T\_XS field of this register. Based on this calculation, a value of 6h should be written to SDSRETR. Figure 17-29 shows how SDSRETR should be programmed.

**Table 17-39. RR Calculation for the EMIF to K4S641632H-TC(L)70 Interface**

Field Name	Formula	Value from K4S641632H-TC(L)70 Datasheet	Value Calculated for Field
T_XS	$T\_XS \geq (t_{XSR} \times f_{EMIF\_CLK}) - 1$	$t_{RC} = 68 \text{ ns (min)}^{(1)}$	6

<sup>(1)</sup> The Samsung datasheet does not specify a  $t_{XSR}$  value. Instead, Samsung specifies  $t_{RC}$  as the minimum required time after CKE going high to complete self refresh exit.

**Figure 17-29. SDRAM Self Refresh Exit Timing Register (SDSRETR)**



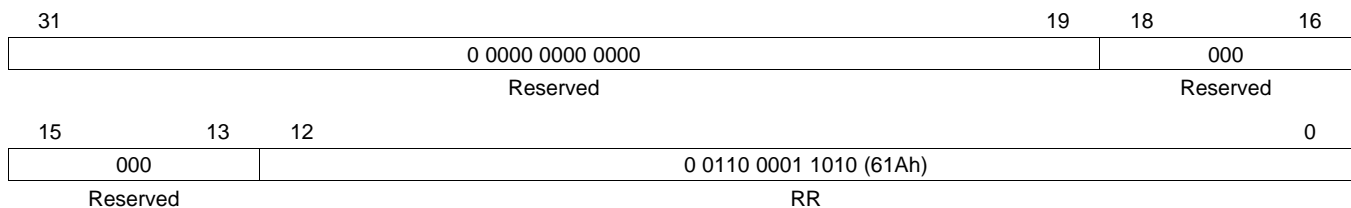
**17.4.2.1.4 SDRAM Refresh Control Register (SDRCR) Settings for the EMIF to K4S641632H-TC(L)70 Interface**

The SDRAM refresh control register (SDRCR) should next be programmed to satisfy the required refresh rate of the K4S641632H-TC(L)70. Table 17-40 shows the calculation of the proper value to program into the RR field of this register. Based on this calculation, a value of 61Ah should be written to SDRRCR. Figure 17-30 shows how SDRRCR should be programmed.

**Table 17-40. RR Calculation for the EMIF to K4S641632H-TC(L)70 Interface**

Field Name	Formula	Values	Value Calculated for Field
RR	$RR \leq f_{EMIF\_CLK} \times t_{Refresh \ Period} / n_{cycles}$	From SDRAM datasheet: $t_{Refresh \ Period} = 64 \text{ ms}$ ; $n_{cycles} = 4096 \text{ EMIF clock}$ rate: $f_{EMIF\_CLK} = 100 \text{ MHz}$	$RR = 1562 \text{ cycles} = 61Ah \text{ cycles}$

**Figure 17-30. SDRAM Refresh Control Register (SDRCR)**



### 17.4.2.1.5 SDRAM Configuration Register (SDCR) Settings for the EMIF to K4S641632H-TC(L)70 Interface

Finally, the fields of the SDRAM configuration register (SDCR) should be programmed as described in [Table 17-37](#) to properly interface with the K4S641632H-TC(L)70 device. Based on these settings, a value of 4720h should be written to SDCR. [Figure 17-31](#) shows how SDCR should be programmed. The EMIF is now ready to perform read and write accesses to the SDRAM.

**Table 17-41. SDCR Field Values For the EMIF to K4S641632H-TC(L)70 Interface**

Field	Value	Purpose
SR	0	To avoid placing the EMIF into the self refresh state
NM	1	To configure the EMIF for a 16-bit data bus
CL	011b	To select a CAS latency of 3
BIT11_9LOCK	1	To allow the CL field to be written
IBANK	010b	To select 4 internal SDRAM banks
PAGESIZE	0	To select a page size of 256 words

**Figure 17-31. SDRAM Configuration Register (SDCR)**

31	30	29	28	24
0	0	0	0 0000	
SR	Reserved	Reserved	Reserved	
23			18	17
		00 0000		0
		Reserved		Reserved
				16
				0
				Reserved
				Reserved
15	14	13	12	11
0	1	0	0	011
Reserved	NM	Reserved	Reserved	CL
				BIT11_9LOCK
7	6		4	3
0		010	0	2
Reserved		IBANK	Reserved	PAGESIZE
				0
				000

### 17.4.2.2 Configuring the Flash Interface

This section describes how to configure the EMIF to interface with the two of SHARP LH28F800BJE-PTTL90 8Mb Flash memory with a clock frequency of  $f_{EMIF\_CLK} = 100$  MHz. The example assumes that one flash is connected to EMIF\_nCS2 and the other to EMIF\_nCS3.

#### 17.4.2.2.1 Asynchronous 1 Configuration Register (CE2CFG) Settings for the EMIF to LH28F800BJE-PTTL90 Interface

The asynchronous 1 configuration register (CE2CFG) and asynchronous 2 configuration register (CE3CFG) are the only registers that is necessary to program for this asynchronous interface (assuming that one Flash is connected to EMIF\_nCS[2] and the other to EMIF\_nCS[3]). The SS bit (in both registers) should be set to 1 to enable Select Strobe Mode and the ASIZE field (in both registers) should be set to 1 to select a 16-bit interface. The other fields in this register control the shaping of the EMIF signals, and the proper values can be determined by referring to the AC Characteristics in the Flash datasheet and the device datasheet. Based on the following calculations, a value of 8862 25BDh should be written to CE2CFG. Table 17-42 and Table 17-43 show the pertinent AC Characteristics for reads and writes to the Flash device, and Figure 17-32 and Figure 17-33 show the associated timing waveforms. Finally, Figure 17-34 shows programming the CE $n$ CFG ( $n = 2, 3$ ) with the calculated values.

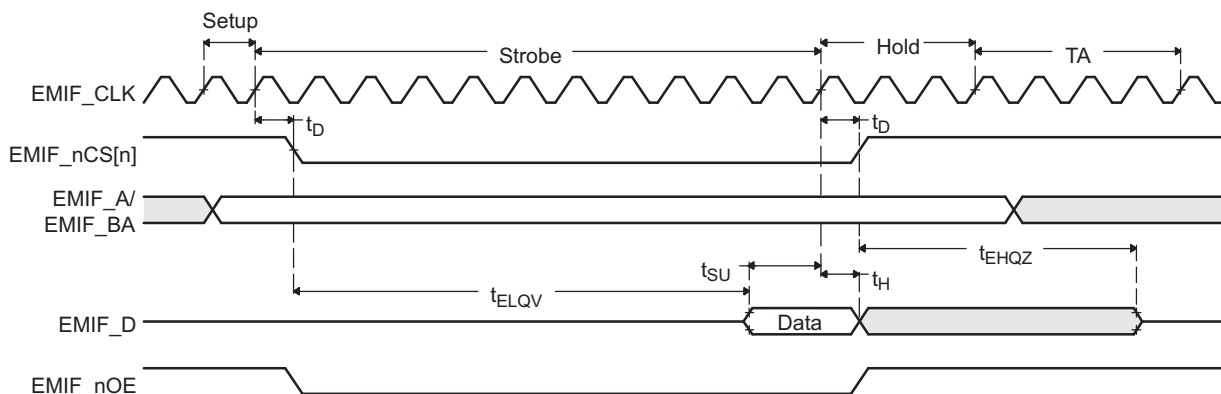
**Table 17-42. AC Characteristics for a Read Access**

AC Characteristic	Device	Definition	Min	Max	Unit
$t_{SU}$	EMIF	Setup time, read EMIF_D before EMIF_CLK high	6.5		ns
$t_H$	EMIF	Data hold time, read EMIF_D after EMIF_CLK high	1		ns
$t_D$	EMIF	Output delay time, EMIF_CLK high to output signal valid		7	ns
$t_{ELQV}$	Flash	nCE to Output Delay		90	ns
$t_{EHQZ}$	Flash	nCE High to Output in High Impedance		55	ns

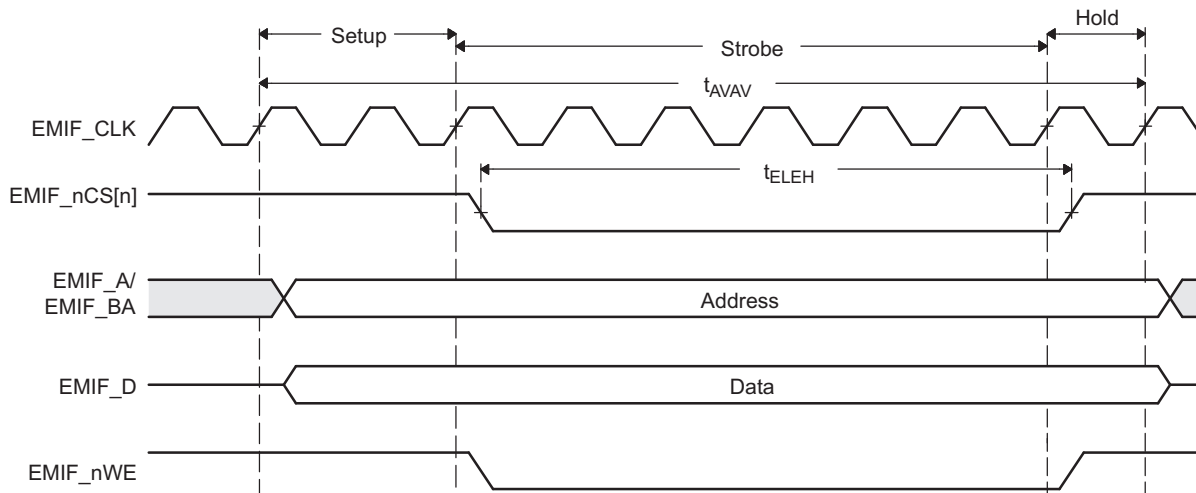
**Table 17-43. AC Characteristics for a Write Access**

AC Characteristic	Device	Definition	Min	Max	Unit
$t_{AVAV}$	Flash	Write Cycle Time	90		ns
$t_{ELEH}$	Flash	nCE Pulse Width Low	50		ns
$t_{EHEL}$	Flash	nCE Pulse Width High (not shown in Figure 17-33)	30		ns

**Figure 17-32. LH28F800BJE-PTTL90 to EMIF Read Timing Waveforms**





**Figure 17-33. LH28F800BJE-PTTL90 to EMIF Write Timing Waveforms**


The R\_STROBE field should be set to meet the following equation:

$$R\_STROBE \geq (t_D + t_{ELQV} + t_{SU}) \times f_{EMIF\_CLK} - 1$$

$$R\_STROBE \geq (7 \text{ ns} + 90 \text{ ns} + 6.5 \text{ ns}) \times 100 \text{ MHz} - 1$$

$$R\_STROBE \geq 9.35$$

$$R\_STROBE = 10$$

The R\_HOLD field must be large enough to satisfy the EMIF Data hold time,  $t_H$ :

$$R\_HOLD \geq t_H \times f_{EMIF\_CLK} - 1$$

$$R\_HOLD \geq 1 \text{ ns} \times 100 \text{ MHz} - 1$$

$$R\_HOLD \geq -0.9$$

The R\_HOLD field must also combine with the TA field to satisfy the Flash's nCE High to Output in High Impedance time,  $t_{EHQZ}$ :

$$R\_HOLD + TA \geq (t_D + t_{EHQZ}) \times f_{EMIF\_CLK} - 2$$

$$R\_HOLD + TA \geq (7 \text{ ns} + 55 \text{ ns}) \times 100 \text{ MHz} - 2$$

$$R\_HOLD + TA \geq 4.2$$

The largest value that can be programmed into the TA field is 3h, therefore the following values can be used:

$$R\_HOLD = 2$$

$$TA = 3$$

For Writes, the W\_STROBE field should be set to satisfy the Flash's nCE Pulse Width constraint,  $t_{ELEH}$ :

$$W\_STROBE \geq t_{ELEH} \times f_{EMIF\_CLK} - 1$$

$$W\_STROBE \geq 50 \text{ ns} \times 100 \text{ MHz} - 1$$

$$W\_STROBE \geq 4$$

The W\_SETUP and W\_HOLD fields should combine to satisfy the Flash's nCE Pulse Width High constraint,  $t_{EHEL}$ , when performing back-to-back writes:

$$W\_SETUP + W\_HOLD \geq t_{EHEL} \times f_{EMIF\_CLK} - 2$$

$$W\_SETUP + W\_HOLD \geq 30 \text{ ns} \times 100 \text{ MHz} - 2$$

$$W\_SETUP + W\_HOLD \geq 1$$

In addition, the entire Write access length must satisfy the Flash's minimum Write Cycle Time,  $t_{AVAV}$ :

$$W\_SETUP + W\_STROBE + W\_HOLD \geq t_{AVAV} \times f_{EMIF\_CLK} - 3$$

$$W\_SETUP + W\_STROBE + W\_HOLD \geq 90 \text{ ns} \times 100 \text{ MHz} - 3$$

$$W\_SETUP + W\_STROBE + W\_HOLD \geq 6$$

Solving the above equations for the Write fields results in the following possible solution:

$$W\_SETUP = 1$$

$$W\_STROBE = 5$$

$$W\_HOLD = 0$$

Adding a 10 ns (1 cycle) margin to each of the periods (excluding TA which is already at its maximum) in this example produces the following recommended values:

$$W\_SETUP = 2h$$

$$W\_STROBE = 6h$$

$$W\_HOLD = 1h$$

$$R\_SETUP = 1h$$

$$R\_STROBE = 6h$$

$$R\_HOLD = 3h$$

$$TA = 3h$$

**Figure 17-34. Asynchronous  $m$  Configuration Register ( $m = 1, 2$ ) (CE $n$ CFG ( $n = 2, 3$ ))**

	31		30		29		26		25		24									
	1		0	0010				00												
	SS		EW	W_SETUP				W_STROBE												
					23		20		19		17		16							
	0110				001				0											
	W_STROBE				W_HOLD				R_SETUP											
		15		13		12		7		6		4		3		2		1		0
	001			001011			011			11		01								
	R_SETUP			R_STROBE			R_HOLD			TA		ASIZE								



## **Analog To Digital Converter (ADC) Module**

---



---

This chapter describes the analog to digital converter (ADC) module.

Topic	Page
<b>18.1 Overview .....</b>	<b>802</b>
<b>18.2 Introduction .....</b>	<b>803</b>
<b>18.3 Basic Features and Usage of the ADC .....</b>	<b>806</b>
<b>18.4 Advanced Conversion Group Configuration Options .....</b>	<b>813</b>
<b>18.5 ADC Module Basic Interrupts .....</b>	<b>817</b>
<b>18.6 ADC Module DMA Requests .....</b>	<b>818</b>
<b>18.7 ADC Magnitude Threshold Interrupts .....</b>	<b>819</b>
<b>18.8 ADC Special Modes .....</b>	<b>820</b>
<b>18.9 ADC Results' RAM Special Features .....</b>	<b>827</b>
<b>18.10 ADEVT Pin General Purpose I/O Functionality .....</b>	<b>829</b>
<b>18.11 ADC Control Registers .....</b>	<b>831</b>

## 18.1 Overview

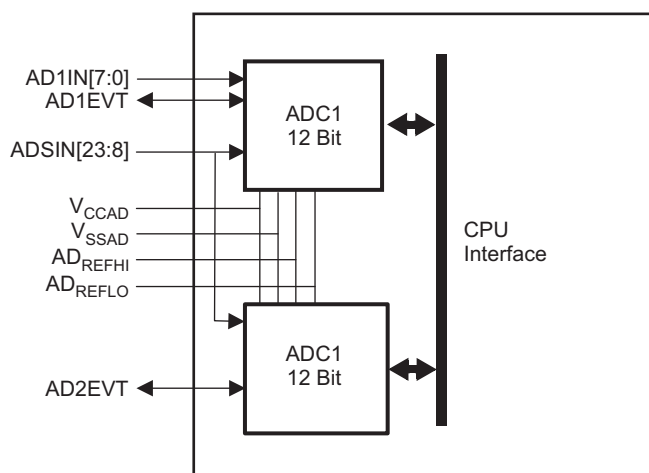
The microcontroller includes two 12-bit ADC modules.

The main features of each of the ADC modules are:

- Selectable 10-bit or 12-bit resolution
- Successive-approximation-register architecture
- Three conversion groups – Group1, Group2 and Event Group
- All three conversion groups can be configured to be hardware-triggered; group1 and group2 can also be triggered by software
- Conversion results are stored in a 64-word memory (SRAM)
  - These 64 words are divided between the three conversion groups and are configurable by software
  - Accesses to the conversion result RAM are protected by parity
- Flexible options for generating DMA requests for transferring conversion results
- Multichannel conversions performed in ascending order, one channel at a time
- Single or continuous conversion modes
- Embedded self-test logic for input channel failure mode (open / short) detection
- Embedded calibration logic for offset error correction
- Enhanced Power-down mode
- External event pin (ADEVT) to trigger conversions
  - ADEVT is also programmable as general-purpose I/O
- Eight hardware events to trigger conversions

The two instances of the 12-bit ADC modules on the microcontroller share 16 analog input channels. The connections are shown below.

**Figure 18-1. Channel Assignments of Two ADC cores**

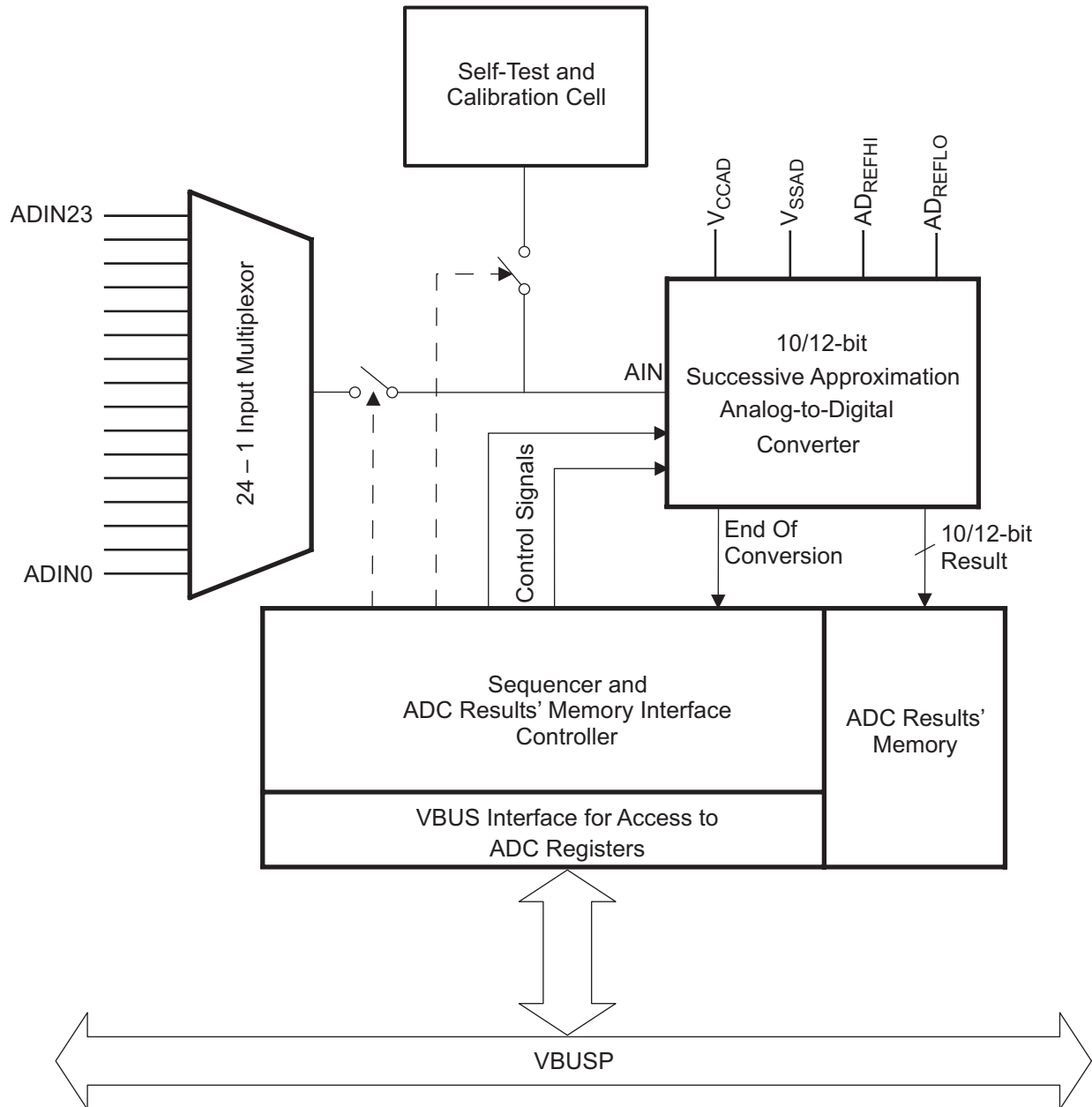


- ADC1 supports 24 channels
- ADC2 supports 16 channels, all of which are shared with ADC1
- The reference voltages as well as operating supply and ground are shared between the two ADC cores

## 18.2 Introduction

This section presents a brief functional description of the analog-to-digital converter (ADC) module. [Figure 18-2](#) illustrates the components of the ADC module.

**Figure 18-2. ADC Block Diagram**



### 18.2.1 Input Multiplexor

The input multiplexor (MUX) connects the selected input channel to the AIN input of the ADC core. The ADC1 module supports up to 24 inputs as shown in [Figure 18-2](#). The ADC2 module supports up to 16 inputs. The sequencer selects the channel to be converted.

### 18.2.2 Self-Test and Calibration Cell

The ADC includes specific hardware that allows a software algorithm to detect open/short on an ADC analog input. It also allows the application program to calibrate the ADC. Also see [Section 18.8.1](#) and [Section 18.8.2](#).

### 18.2.3 Analog-to-Digital Converter Core

The ADC core is a combination voltage scaling, charge redistribution Successive Approximation Register (SAR) based analog-to-digital converter. The core can be configured for operation in 10-bit resolution (default) or 12-bit resolution. This is controlled by the sequencer logic. This selection applies to all conversions performed by the ADC module. It is not possible to convert some channels with a 12-bit resolution and some with a 10-bit resolution.

A single conversion from an analog input to a digital conversion result occurs in two distinct periods:

- Sampling Period:
  - The sequencer generates a START signal to the ADC core to signal the start of the sampling period.
  - The analog input signal is sampled directly on to the switched capacitor array during this period, providing an inherent sample-and-hold function.
  - The sampling period ends one full ADCLK after the falling edge of the START signal.
  - The sequencer can control the sampling period duration by configuring the conversion group's sample time control register (ADEVSAMP, ADG1SAMP, ADG2SAMP). This register controls the time for which the START signal stays high.
- Conversion Period:
  - The conversion period starts one full ADCLK after the falling edge of START.
  - One bit of the conversion result is output on each rising edge of ADCLK in the conversion period, starting with the most-significant bit first.
  - The conversion period is 12 ADCLK cycles in case of a 12-bit ADC, and is 10 ADCLK cycles in case of a 10-bit ADC.
  - The ADC core generates an End-Of-Conversion (EOC) signal to the sequencer at the end of the conversion period. At this time the complete 12-, or 10-bit conversion result is available.
  - The sequencer captures the ADC core conversion result output as soon as EOC is driven High.

The analog conversion range is determined by the reference voltages:  $AD_{REFHI}$  and  $AD_{REFLO}$ .  $AD_{REFHI}$  is the top reference voltage and is the maximum analog voltage that can be converted. An analog input voltage equal to  $AD_{REFHI}$  results in an output code of 0x3FF for 10-bit resolution and 0xFFF for 12-bit resolution.  $AD_{REFLO}$  is the bottom reference voltage and is the minimum analog voltage that can be converted. Applying an input equal to  $AD_{REFLO}$  results in an output code of 0x000. Both  $AD_{REFHI}$  and  $AD_{REFLO}$  must be chosen not to exceed the analog power supplies:  $V_{CCAD}$  and  $V_{SSAD}$  respectively. Input voltages between  $AD_{REFHI}$  and  $AD_{REFLO}$  produce a conversion result given by [\(Equation 27\)](#) for 10-bit resolution and [\(Equation 28\)](#) for 12-bit resolution.

$$DigitalResult = \frac{1024 \times (InputVoltage - AD_{REFLO})}{AD_{REFHI} - AD_{REFLO}} - 0.5 \quad (27)$$

$$DigitalResult = \frac{4096 \times (InputVoltage - AD_{REFLO})}{(AD_{REFHI} - AD_{REFLO})} - 0.5 \quad (28)$$

### 18.2.4 Sequencer

The sequencer coordinates the operations of the ADC, including the input multiplexor, the ADC core, and the result memory. In addition, the logic of the sequencer sets the status register flags when the conversion is ongoing, stopped, or finished.

All the features of the sequencer are discussed in detail in the following sections of this document.

### 18.2.5 Conversion Groups

Several applications require groups of channels to be converted using a single trigger source for example. There could also be some groups of channels identified which require a specific setting of the acquisition time. The ADC module supports three conversion groups for this purpose – Group1, Group2 and the Event Group.

Any of the available analog input channels can be assigned to any of the conversion groups. This also allows a particular channel to be repeatedly sampled by selecting it in multiple groups. There is an inherent priority scheme used when multiple conversion groups are triggered at once. The Event Group is the highest-priority, followed by the Group1 and then the Group2.

The Event Group is always hardware event-triggered. Group1 and Group2 are software-triggered by default and can be configured to be hardware-, or event-triggered as well. The triggering of conversions in each group is discussed in [Section 18.3.6](#).

Each conversion group has a separate set of control registers to:

- Select the input channels to be converted
- Configure the mode of conversion: single conversion sequence or continuous conversions
- Configure the input channel sampling time
- Configure the interrupt and/or DMA request generation conditions



## 18.3 Basic Features and Usage of the ADC

This section describes the usage of the basic features of the ADC module.

### 18.3.1 How to Select Between 12-bit and 10-bit Resolutions

The 10/12-BIT field of the ADC Operating Mode Control Register (ADOPMODECR) configures the ADC to be in 10-bit or 12-bit resolution mode.

- If 10/12-BIT = 0, the module is in 10-bit resolution mode. This is the default mode of operation.
- If 10/12-BIT = 1, the module is in 12-bit resolution mode.

### 18.3.2 How to Setup the ADCLK Speed

The ADC sequencer generates the clock for the ADC core, ADCLK. The ADC core uses the ADCLK signal for its timing. The ADCLK is generated by dividing down the input clock to the ADC module, which is the VBUSP interface clock, VCLK. A 5-bit field (PS) in the ADC Clock Control Register (ADCLOCKCR) is used to divide down the VCLK by 1 up to 32. The ADCLK valid frequency range is specified in the device datasheet.

$$f_{\text{ADCLK}} = f_{\text{VCLK}} / (\text{PS} + 1)$$

The maximum frequency for ADCLK is specified in the device datasheet.

### 18.3.3 How to Setup the Input Channel Acquisition Time

The signal acquisition time for each group is separately configurable using the ADG1SAMP[11:0], ADG2SAMP[11:0], and ADEVSAMP[11:0] registers.

The acquisition time is specified in terms of ADCLK cycles and ranges from a minimum of 2 ADCLK cycles to a maximum of 4098 ADCLK cycles.

For example, Group1 acquisition time,  $t_{\text{ACQG1}} = \text{G1SAMP}[11:0] + 2$ , in ADCLK cycles.

The minimum acquisition time is specified in the device datasheet. This time also depends on the impedance of the circuit connected to the analog input channel being converted.

### 18.3.4 How to Select an Input Channel for Conversion

The ADC module needs to be enabled first before selecting an input channel for conversion. The ADC module can be enabled by setting the ADC EN bit in the ADC Operating Mode Control Register (ADOPMODECR). Multiple input channels can be selected for conversion in each group. Only one input channel is converted at a time. The channels to be converted are configured in one or more of the three conversion groups' channel selection registers. Channels to be converted in Group1 are configured in the Group1 Channel-Select Register (ADG1SEL), those to be converted in Group2 are configured in the Group2 Channel-Select Register (ADG2SEL), and those to be converted in the Event Group are configured in the Event Group Channel-Select Register (ADEVSEL).

### 18.3.5 How to Select Between Single Conversion Sequence or Continuous Conversions

Each group has its own mode control register. The CONV MODE field of these control registers allow the application to select between a single conversion sequence or continuous conversion mode.

---

**NOTE: Selecting continuous conversion mode for all three groups**

All three conversion groups cannot be configured to be in a continuous conversion mode. If the application configures the group mode control registers to enable continuous conversion mode for all three groups, then the Group2 will be automatically be configured to be in a single conversion sequence mode.

---

With conversions ongoing in continuous conversion mode, if the CONV MODE field of a group is cleared, then that group switches to the single conversion sequence mode. Conversions for this group will stop once all channels selected for that group have been converted.

### **18.3.6 How to Start a Conversion**

The conversion groups Group1 and Group2 are software-triggered by default. A conversion in these groups can be started just by writing the desired channels to the respective Channel-Select Registers. For example, in order to convert channels 0, 1, 2, and 3 in Group1 and channels 8, 9, 10, and 11 in Group2, the application just has to write 0x0000000F to ADG1SEL and 0x00000F00 to ADG2SEL. The ADC module will start by servicing the group that was triggered first, Group1 in this example.

The conversions for all groups are performed in ascending order of the channel number. For the Group1 the conversions will be performed in the order: channel 0 first, followed by channel 1, then channel 2, and then channel 3. The Group2 conversions will be performed in the order: channels 8, 9, 10, and 11.

The Event Group is only hardware-triggered. There are up to eight hardware event trigger sources defined for the ADC module. Check the device datasheet for a complete listing of these eight hardware trigger options.

The trigger source to be used needs to be configured in the ADEVSRC register. Similar registers also exist for the Group1 and Group2 as these can also be configured to be event-triggered.

The polarity of the event trigger is also configurable, with a falling edge being the default.

An Event Group conversion starts when at least one channel is selected for conversion in this group, and when the defined event trigger occurs.

If any conversion group is configured to be in a continuous conversion mode, then it needs to only be triggered once. All the channels selected for conversion in that group will be converted repeatedly.

### **18.3.7 How to Know When the Group Conversion is Completed**

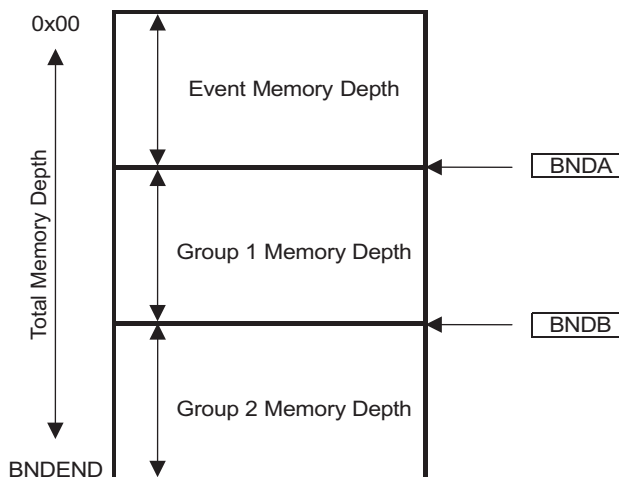
Each conversion group has a status flag to indicate when its conversion has ended. See ADEVSR, ADG1SR and ADG2SR. This bit is set when a conversion sequence for a group ends. This bit does is always set if a group is configured for continuous conversions.

### **18.3.8 How Results are Stored in the Results' Memory**

The ADC stores the conversion results in three separate memory regions in the ADC Results' RAM, one region for each group. Each memory region is a stack of buffers, with each buffer capable of holding one conversion result. The number of buffers allocated for each group is programmed by configuring the ADC module registers ADBNDCR and ADBNDEND.

ADBNDCR contains two 9-bit pointers BNDA and BNDB. BNDA, BNDB and BNDEND are used to partition the total memory available into three memory regions as shown in [Figure 18-3](#). Both BNDA and BNDB are pointers referenced from the start of the results' memory. BNDA specifies the number of buffers allocated for the Event Group conversion results in units of two buffers; BNDB specifies the number of buffers allocated for the Event Group plus Group1 in units of two buffers. Please refer to [Section 18.11.23](#) for more details on configuring the ADC results' memory.

ADBNDEND contains a 3-bit field called BNDEND that configures the total memory available. The ADC module can support up to 1024 buffers. The device supports a maximum of 64 buffers for both the ADC modules.

**Figure 18-3. FIFO Implementation**

- Number of buffers for Event Group = 2 x BNDA
- Number of buffers for Group1 = 2 x (BNDB – BNDA)
- Number of buffers for Group2 = Total number of buffers – 2 x BNDB

### 18.3.9 How to Read the Results from the Results' Memory

The CPU can read the conversion results in one of two ways:

- By using the conversion results memory as a FIFO queue, or
- By accessing the conversion results memory directly

#### 18.3.9.1 Reading Conversion Results from a FIFO

The conversion results for each group can be accessed via a range of addresses provided to facilitate the use of the ARM Cortex-R4 CPU's Load-Multiple (LDM) instruction. A single read performed using the LDR instruction can also be used to read out a single conversion result. The results are read out from the group's memory region as a FIFO queue by reading from any location inside this address range. The conversion result that got stored first gets read first. A result that is read from the memory in this method is removed from the memory. For example, a read from any address in the range ADEVBUFFER (offset 0x90 to 0xAF) pulls out one conversion result from the Event Group memory.

**Figure 18-4. Format of Conversion Result Read from FIFO, 12-bit ADC**

Offset Address Register	31	15	30	14	29	13	28	12	27	11	26	10	25	9	24	8	23	7	22	6	21	5	20	4	19	3	18	2	17	1	16	0
0x90 to 0xAFADEVBUFFER	EV EMP TY	Reserved																				EV_CHID										
		Reserved										EV_DR																				
0xB0 to 0xCFADG1BUFFER	G1 EMP TY	Reserved																				G2_CHID										
		Reserved										G1_DR																				
0xD0 to 0xEFADG2BUFFER	G2 EMP TY	Reserved																				G2_CHID										
		Reserved										G2_DR																				

**Figure 18-5. Format of Conversion Result Read from FIFO, 10-bit ADC**

Offset Address Register	31 15	30 14	29 13	28 12	27 11	26 10	25 9	24 8	23 7	22 6	21 5	20 4	19 3	18 2	17 1	16 0
0x90 to 0xAFADDEVBUFFER	EV EMP TY	EV_CHID						EV_DR								
0xB0 to 0xCFADG1BUFFER	EV EMP TY	G1_CHID						G1_DR								
0xD0 to 0xEFADG2BUFFER	EV EMP TY	G2_CHID						G2_DR								

**Option to read channel id along with conversion result:**

The application has an option to read the channel id along with the conversion result. This is controlled by the CHID field of the group’s mode control register. If the option to read the channel id is not selected, the channel id field of the conversion result reads as zeros.

**Protection against reading from empty FIFO:**

There is also a hardware mechanism to protect the application from reading past the number of new conversion results held in the FIFO. Once all available conversion results have been read out of the FIFO by the application, a subsequent read from the FIFO causes the mechanism to indicate that the FIFO is empty by setting the EMPTY field.

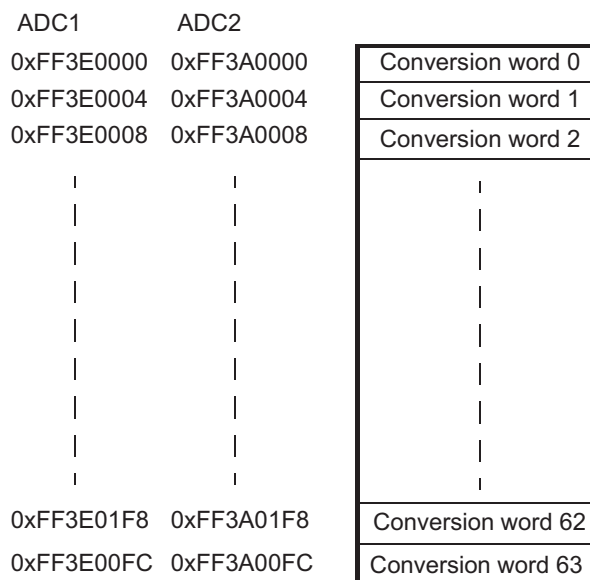
**Debug / Emulation Support:**

For debug purposes, each conversion group also provides an address that the application can read from for extracting the group’s conversion results. However, no status flags for a conversion group are affected by reading from these emulation buffer addresses. For example, reading from ADEVEMUBUFFER (offset 0xF0) returns the next result in the Event Group buffer but does not actually remove that result from the buffer or change the amount of data held in the buffer.

**18.3.9.2 Reading Conversion Results Directly from the Conversion Results’ Memory**

The conversion result memory is part of the device’s memory map. The base address for the ADC1 result memory is 0xFF3E0000, while that for the ADC2 result memory is 0xFF3A0000.

**Figure 18-6. ADC Memory Mapping**



The application can identify the address ranges for each of the three memory regions for the three conversion groups after performing the segmentation as described in [Section 18.3.8](#). It is up to the application to read the desired results from the three conversion groups. The formats of the conversion results when reading from RAM directly are shown below.

**Figure 18-7. Format of Conversion Result Directly Read from ADC RAM, 12-bit ADC**

ADC RAM addresses	Reserved		channel id [4]
	channel id [3–0]	12-bit conversion result	

**Figure 18-8. Format of Conversion Result Directly Read from ADC RAM, 10-bit ADC**

ADC RAM addresses	Reserved		
	Reser ved	channel id [4–0]	10-bit conversion result

Note that there is no EMPTY field to protect the application from reading data that has been previously read.

Each group does have a separate register which holds the address in the group's result memory where the ADC will write the next conversion result. These are the ADEVRAMWRADDR, ADG1RAMWRADDR and ADG2RAMWRADDR registers. The application can use this information to calculate how many valid conversion results are available to be read.

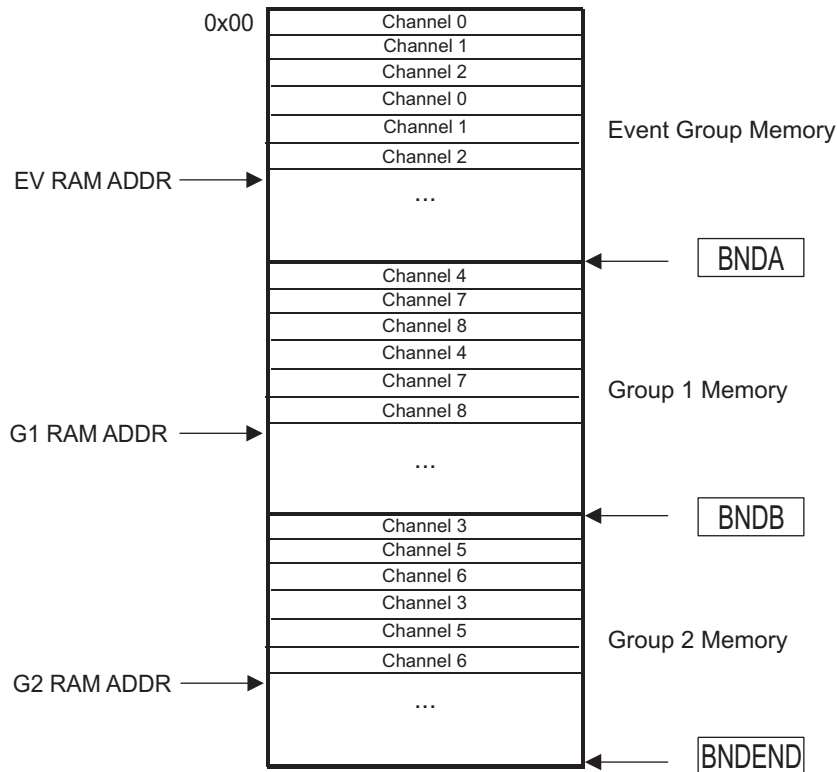
***Benefit of reading conversion results directly from ADC RAM:***

The application does not have to read out conversion results sequentially as in the case of reading from a FIFO. As a result, the application can selectively read the conversion results for any particular input channel of interest without having to read other channels' conversion results.

### 18.3.9.3 Example

Suppose that channels 0, 1, and 2 are selected for conversion in the Event Group, channels 4, 7, and 8 are selected for conversion in group 1, and channels 3, 5, and 6 are selected for conversion in group 2. The conversion results will get stored in the three memory regions as shown below:

**Figure 18-9. 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 ADEVBUFFER (offset 0x90 to 0xAF) multiple times, or do a “load multiple” from this range of addresses. This will cause the ADC to return the results for channel 0, then channel 1, then channel 2, then channel 0, and so on for each read access to this address range.

Now suppose that the application wants to read out the results for the group 1 from the RAM directly. The conversion results for the group 1 are accessible starting from address ADC RAM Base Address + BNDA. Also, it is known that the first result at this address is for the input channel 4, the next one is for input channel 7, and so on. So the application can selectively read the conversion results for only one channel if so desired.

### 18.3.10 How to Stop a Conversion

A group’s conversion can be stopped by clearing the group’s channel select register.

### 18.3.11 Example Sequence for Basic Configuration of ADC Module

The following sequence is necessary to configure the ADC to convert channels 0, 2, 4, and 8 in single-conversion mode using Group1:

1. Write 0 to the Reset Control Register (ADRSTCR) to release the module from the reset state
2. Write 1 to the ADC EN bit of the Operating Mode Control Register (ADOPMODECR) to enable the ADC state machine
3. Configure the ADCLK frequency by programming the desired divider into the Clock Control Register (ADCLOCKCR)
4. Configure the acquisition time for the group that is to be used. For example, configure the Group1 Sampling Time Control Register (ADG1SAMP) to set the acquisition time for Group1.
5. Select the channels that need to be converted in Group1 by writing to the Group1 Channel Select Register (ADG1SEL). In this example, a value of 0x115 needs to be written to ADG1SEL in order to select channels 0, 2, 4, and 8 for conversion in Group1.

- 
- The ADC sequencer will start the Group1 conversions as soon as the write to the ADG1SEL register is completed.
6. Wait for the GP1 END bit to be set in the Group1 Conversion Status Register (ADG1SR). This bit gets set when all the channels selected for conversion in Group1 are converted and the results are stored in the Group1 memory.
  7. Read the conversion results by reading from the Group1 FIFO access location (ADG1BUFFER) or by reading directly from the Group1 results' memory.

## 18.4 Advanced Conversion Group Configuration Options

The below [Table 18-1](#) shows the operating mode control registers and the status registers for each of the three conversion groups. The register addresses shown are offsets from the base address. The ADC1 register frame base address is 0xFFF7C000 and the ADC2 register frame base address is 0xFFF7C200.

**Table 18-1. ADC Groups' Operating Mode Control and Status Registers**

Offset Address Register	31 15	30 14	29 13	28 12	27 11	26 10	25 9	24 8	23 7	22 6	21 5	20 4	19 3	18 2	17 1	16 0
0x010ADEVM ODECR	Reserved															No Reset On Chn Sel
	Reserved						EV_DATA_ FMT	Reserved	EV CHID	OVR EV RAM IGN	Rese rved	EV 8BIT	EVM ODE	FRZ EV		
0x014ADG1M ODECR	Reserved															No Reset On Chn Sel
	Reserved						G1_DATA_ FMT	Reserved	G1 CHID	OVR G1 RAM IGN	G1 HW TRIG	G1 8BIT	G1M ODE	FRZ G1		
0x018ADG2M ODECR	Reserved															No Reset On Chn Sel
	Reserved						G2_DATA_ FMT	Reserved	G2 CHID	OVR G2 RAM IGN	G2 HW TRIG	G2 8BIT	G2M ODE	FRZ G2		
0x06CADEVSR	Reserved															
	Reserved											EV MEM EMP TY	EV BUS Y	EV STO P	EV END	
0x070ADG1SR	Reserved															
	Reserved											G1 MEM EMP TY	G1 BUS Y	G1 STO P	G1 END	
0x074ADG2SR	Reserved															
	Reserved											G2 MEM EMP TY	G2 BUS Y	G2 STO P	G2 END	

The following sections describe each of these group configuration options separately.

### 18.4.1 Group Trigger Options

The Group1 and Group2 operating mode control registers have an extra control bit: HW TRIG. This bit configures the group to be hardware event-triggered instead of software-triggered, which is the default.



When a group is configured to be event-triggered, the group conversion starts when at least one channel is selected for conversion in this group, and when the defined event trigger occurs. The event trigger source is defined for each group in the ADEVSRC, ADG1SRC and the ADG2SRC registers. The actual connections used as the event trigger sources are defined in the device datasheet for both the ADC modules.

### 18.4.2 Single or Continuous Conversion Modes

The EV MODE, G1 MODE, and G2 MODE bits are used to select between either single or continuous conversion mode for each of the three groups.

#### 18.4.2.1 Single Conversion Mode

A conversion group configured to be in single-conversion mode gets serviced only once by the ADC for each group trigger. The trigger can be a software trigger as in the case of Group1 and Group2 by default, or it could be a hardware event trigger as in the case of the Event Group or Group1 or Group2.

The entire conversion sequence, from the acceptance of the group conversion request to the end of the last channel's conversion, is flagged for each group by the corresponding BUSY bit in that group's status register. After single-conversion mode is started, the BUSY bit is read as 1 until the conversion of the last channel is complete. The END bit for the group is set once all the channels in that group are converted.

For example, say channels 0, 2, 4, and 6 are selected for conversion in Group1 in single-conversion mode. When the Group1 gets serviced, the ADC will start conversion for channel 0, then channel 2, then channel 4, and then channel 6. It will then stop servicing the Group1, set the GP1 END status bit, and look to service the Event Group or the Group2, if required.

#### 18.4.2.2 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.

---

### 18.4.3 Conversion Group Freeze Capability

The ADC module has an inherent priority order between the three conversion groups. This group priority determines the order of conversion in case multiple groups are triggered. The priority of conversions between the three groups in descending order is:

1. Event Group
2. Group1
3. Group2

Examples of conversion group priority:

- If an Event Group conversion is ongoing in single conversion sequence mode and Group2 and Group1 conversions are requested, then the ADC will finish conversion of channels selected in Event Group, then switch over to converting channels selected in Group1, and then convert channels selected in Group2.
- If Group1 conversions are ongoing in continuous conversion mode and Group2 conversion is requested, then the ADC will complete converting the current channel for Group1 and switch over to converting channels selected in Group2. The new conversion request for Group2 has a higher priority than the pending continuous conversion request for Group1.

The conversion group freeze capability allows the application to override this default priority between the conversion groups. Enabling the freeze capability allows the ADC to freeze a higher-priority conversion group's conversions whenever there is a request for conversion in another (lower-priority) group.

For example, setting the FRZ EV bit in the ADEVMODECR register will allow the ADC to freeze ongoing Event Group conversions whenever there is a pending request, or a new request for a Group1 or Group2 conversion. The conversions for the Event Group will be frozen as long as the Group1 or Group2 conversions are active. Once the Group1 or Group2 conversions are completed, the Event Group conversions start from where they were frozen.

While a group's conversions are frozen, the group's STOP status bit is set. This bit is cleared once the group's conversions are restarted.

#### **18.4.4 Conversion Group Memory Overrun Option**

An overrun condition occurs when the ADC module tries to store more conversion results to a group's results' memory which is already full. In this case, the ADC allows two options.

If the OVR RAM IGN bit in the group's operating mode control register (ADEVMODECR, ADG1MODECR, ADG1MODECR) is set, then the ADC module ignores the contents of the group's results' memory and wraps around to overwrite the memory with the results of new conversions.

If the OVR RAM IGN bit is not set, then the application program has to read out the group's results' memory upon an overrun condition; only then can the ADC continue to write new results to the memory.

#### **18.4.5 Response on Writing Non-Zero Value to Conversion Group's Channel Select Register**

If the application writes a non-zero value to a group's channel select register while that group's conversions are already being serviced, then that group's conversions will be restarted with the new configuration programmed in the channel select registers.

The following rules apply in terms of the effect on the ADC conversion sequence:

- If the new conversion request comes from the same group as the ongoing conversion, then the ongoing conversion will be stopped in whichever stage it is in, and the new sequence of conversions will be started.
- If the new conversion request comes from a separate group, then the ongoing channel's conversion will be completed before starting the new sequence of conversions.

The following rules apply in terms of the effect on the group's results memory:

- If a group conversion is ongoing or is frozen, writing a non-zero value to the group's channel select register will also reset its results FIFO. This does not clear the contents of the results FIFO; only the ADC module is allowed to overwrite the FIFO's contents with new conversion results starting from the first location.
- If the group conversion is completed (CONV <GRP> END flag is set), or the group is not being used, then writing a non-zero value to the group's channel select register will either be reset or not depending on the value of the NoResetOnChnSel bit for that group (ADEVMODECR, ADG1MODECR, ADG1MODECR).
  - If the NoResetOnChnSel bit is 0, then the group's FIFO will be reset.
  - If the NoResetOnChnSel bit is 1, then the group's FIFO will not be reset.

### 18.4.6 Conversion Result Size on Reading: 8-bit, 10-bit or 12-bit

Some applications don't need the full 12-bit resolution of the ADC modules on the TMS570LS30336U, and can work with 8-bit or 10-bit conversion results.

#### 18.4.6.1 ADC Configured in 12-bit Resolution

The mode control register for each conversion group contains a field called DATA\_FMT, which defines the format of the conversion result read out of the result RAM, when accessed as a FIFO.

The DATA\_FMT field is encoded as follows:

- If DATA\_FMT = 00, the complete 12-bit conversion result is read out of the FIFO.
- If DATA\_FMT = 01, the 12-bit conversion result is right-shifted by 2 and the resulting 10-bit result is read out of the FIFO.
- If DATA\_FMT = 10, the 12-bit conversion result is right-shifted by 4 and the resulting 8-bit result is read out of the FIFO.

This control field is not effective when the application chooses to access the conversion result memory directly. In that case, the application can choose to mask off the number of bits as required.

#### 18.4.6.2 ADC Configured in 10-bit Resolution

The DATA\_FMT field is not effective in this mode and the application has the choice to read either the full 10-bit conversion result or an 8-bit conversion result. This is controlled by the 8BIT field of the group's operating mode control register.

- If 8BIT = 0, the complete 10-bit conversion result is read out of the FIFO.
- If 8BIT = 1, the 10-bit conversion result is right-shifted by 2 and the resulting 8-bit result is read out of the FIFO.

### 18.4.7 Option to Read Group Channel Id Along With Conversion Result

The ADC module allows the application program to also read out the analog input channel number along with its conversion result. This capability is enabled by setting the CHID bit in the group's operating mode control register.

If the CHID bit is not set, the bits [14-10] are forced to 00000 when the conversion results are read out from the group's results' FIFO.

If the CHID bit is set, the bits [14-10] in the group's results' memory contain the input channel number to which the conversion result belongs.

---

#### NOTE: Actual Storage of Channel Id

Regardless of whether the CHID bit is set or not, the channel number is **always stored** in the memory along with the conversion result. The CHID bit only affects whether the channel number is available with the conversion result **when the group's memory is read**. Therefore, the CHID bit for a group can be changed dynamically without affecting that group's ongoing conversions.

---

## **18.5 ADC Module Basic Interrupts**

This section describes the basic interrupts generated by the ADC module.

### **18.5.1 Group Conversion End Interrupt**

The ADC module sets the group's conversion end flag (EV END, G1 END, or G2 END) in that group's interrupt flag register (ADEVINTFLG, ADG1INTFLG, ADG2INTFLG) when all the channels selected for conversion in that group are converted. This causes a group conversion end interrupt to be generated if this interrupt is enabled by setting the group's END INT EN control bit (EV END INT EN, G1 END INT EN, or G2 END INT EN).

This interrupt can be easily used for conversion groups configured to be in the single-conversion mode. The application program can read out the conversion results, change the group's configuration if necessary, and restart the conversions by triggering the group from within the interrupt service routine.

For groups configured to be in continuous conversion mode, this interrupt condition is not practical as the conversions are always in progress. In this case, the Group Memory Threshold Interrupt is more practical as the application can allow a programmable number of conversion results to accumulate before interrupting the CPU.

### **18.5.2 Group Memory Threshold Interrupt**

The ADC module has the ability to generate an interrupt for a fixed number of conversions for each group. A group memory threshold register determines how many conversion results must be in a group's memory region before the CPU is interrupted. This feature can be used to significantly reduce the CPU load when using interrupts for reading the conversion results.

The group's threshold register needs to be configured before the group conversions are triggered. This threshold register value behaves like a down-counter, which decrements each time the ADC writes a conversion result to this group's memory. This counter is incremented each time the application program reads a conversion result from the results' memory by accessing the FIFO queue. Simultaneous read (by application program) and write (by ADC module) operations from the group's results' memory leave the threshold counter unchanged.

The threshold counter can decrement past 0 and become negative. It always increments back to its original value when the memory region is emptied. To determine how many samples are in the memory region at a given moment, the threshold counter can be subtracted from the originally configured threshold count.

Whenever the threshold counter transitions from +1 to 0, it sets the group's threshold interrupt flag, and the CPU is interrupted if the group's threshold interrupt is enabled. The CPU is expected to clear the interrupt flag after reading the conversion results from the memory.

The interrupt flag is not set when the threshold counter stays at 0 or transitions from -1 to 0.

### **18.5.3 Group Memory Overrun Interrupt**

An interrupt can be generated for each group if the number of ADC conversions for that group exceed the number of buffers allocated for that conversion group. The application program can choose to read out all the conversion results using the CPU or the DMA. Alternatively, the application program can set the group's OVR RAM IGN bit and allow the ADC module to overwrite the group's results' memory contents with new conversion results.

## 18.6 ADC Module DMA Requests

This section describes the capabilities of the ADC module to take advantage of the Platform DMA controller module. The ADC module can generate a DMA request under two conditions:

### 18.6.1 DMA request for each conversion result written to the results' memory

In this mode, the ADC module will generate the first DMA request as soon as a conversion result gets written to the group's results' memory. Subsequent writes to the results' memory will cause DMA requests to be generated. This mode allows a smaller amount of ADC results' memory to suffice for an application.

This DMA request generation is enabled by setting the group's DMA EN bit in the group's DMA control register. The BLOCK XFER bit in this register must be left cleared (default) if a DMA request is desired to be generated for new results getting written to the results' memory.

### 18.6.2 DMA request for a fixed number of conversion results

This mode is enabled by setting both, the group's DMA EN, and the group's BLOCK XFER, bits in the group's DMA control registers.

In this mode, a DMA request will be generated for a specified number of conversion results being available in the group's results' memory. The number of conversion results desired are configured using the group's BLOCKS field in the control registers.

For example, if the BLOCK count is configured for 10, then ADC module will generate a DMA request at the end of 10th conversion. DMA controller should complete reading out 10 data before next set of 10 conversions complete.

---

**NOTE: Usage of Block DMA transfers with Threshold Interrupts**

It is not recommended to enable the block DMA transfers for a group at the same time as the group threshold interrupt. The group's BLOCKS field is essentially the same as the group's THRESHOLD field in the group's interrupt control register described in [Section 18.5.2](#).

---

## 18.7 ADC Magnitude Threshold Interrupts

The ADC allows up to three magnitude threshold interrupts to be generated. The comparison parameters are programmed via the Magnitude Threshold Control Register (ADMAGINTxCR).

### 18.7.1 Magnitude Threshold Interrupt Configuration

The following fields are configurable for each of the three available magnitude threshold interrupts:

1. CHN/THR COMP: Specifies whether to compare two channels' conversion results, or to compare a channel's conversion result to a programmable threshold value. A value of 0 will select the programmable threshold to be compared, and a value of 1 will select the conversion result of the channel identified by the COMP CHID field to be compared.
2. MAG CHID: Specifies the channel number from 0 to 23 whose conversion result needs to be monitored.
3. COMP CHID: Specifies the channel number from 0 to 23 whose last conversion result is used for the comparison with the conversion result of the channel being monitored.
4. MAG 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. A value of 0 in the CMP GE/LT field indicates a "less than" comparison and a value of 1 indicates a "greater than or equal to" comparison.

### 18.7.2 Magnitude Threshold Interrupt Comparison Mask Configuration

There is also a separate comparison mask register (ADMAGINTxMASK) for each of the three magnitude threshold interrupts. This register is used to specify the bits that are masked off for the sake of the comparison. For example, the lower 4 bits of the conversion result can be masked off by writing 0xf to the interrupt comparison mask register, allowing a gross comparison to be made. By default, the full 10/12-bit conversion results are compared.

### 18.7.3 Magnitude Threshold Interrupt Enable / Disable Control

Each of the three magnitude interrupts also have separate interrupt enable set (ADMAGTHRINTENASET) and clear (ADMAGTHRINTENACLR) 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.

### 18.7.4 Magnitude Threshold Interrupt Flags

There is a separate Magnitude Interrupt Flag register (ADMAGINTFLG) that holds the flags for these three interrupts. This flag gets set whenever the comparison condition for the corresponding interrupt is met. A magnitude threshold interrupt is generated if the corresponding flag is set inside the flag register, and the interrupt generation is enabled. This flag can be cleared by writing a 1 to the flag or by reading from the interrupt offset register in case of this interrupt being the current highest-priority pending interrupt.

### 18.7.5 Magnitude Threshold Interrupt Offset Register

It is possible to have multiple magnitude threshold interrupts pending at the same time. The magnitude threshold interrupt offset register (ADMAGINTOFF) holds the index of the currently pending highest priority magnitude threshold interrupt. The magnitude threshold interrupt 1 has the highest priority while the magnitude threshold interrupt 3 has the lowest priority. This is a read-only register and returns zeros if none of the magnitude threshold interrupts are pending. Writes to this register have no effect.

A read from this register updates the register to the next highest-priority pending magnitude threshold interrupt. This read also clears the corresponding flag from the magnitude threshold interrupt flag register. However, a read from the magnitude threshold interrupt offset register in emulation mode does not affect the interrupt flag register or the interrupt offset register.



## 18.8 ADC Special Modes

The ADC module supports some special modes for diagnostics and power saving purposes.

### 18.8.1 ADC Error Calibration Mode

The application program can activate a calibration sequence any time self-test mode is disabled (SELF TST = 0). This calibration sequence includes the conversion of an embedded calibration reference voltage followed by the calculation of an offset error correction value.

---

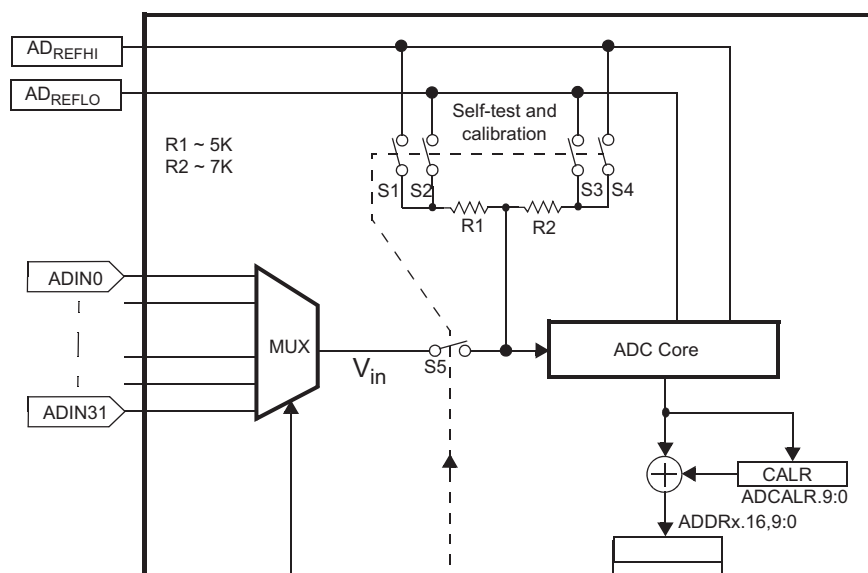
#### NOTE: Disable Self-Test Mode Before Calibration

To avoid errors during the calibration operation, self-test mode must *not* be enabled during a calibration sequence. In addition, to ensure accurate results, calibrate the ADC in an environment with minimum noise.

---

Calibration mode is enabled by setting the CAL EN bit (ADCALCR.0). The application needs to ensure that no conversion group is being serviced when the calibration mode is enabled.

**Figure 18-10. Self-Test and Calibration Logic**



The input multiplexer gets disabled and only the reference voltage is connected to the ADC core input. Switch S5 of Figure 18-10 is opened. In addition, the digital result issued from a conversion is output from the ADC core to the calibration and offset error correction register, ADCALR. The ADC results' memory is not affected by the calibration conversion.

When calibration mode is disabled, the ADC can be configured for normal conversions.

#### 18.8.1.1 Calibration Conversion

The calibration conversion also needs to meet the minimum sampling time specification for the ADC. This value is typically 1  $\mu$ s. The Event Group sample time register (ADEVSAMP) is used to specify the number of ADCLK cycles for the calibration conversion.

The BRIDGE EN and HILO bits (ADCALCR.9:8) control the voltage to the calibration reference device shown in Figure 18-12. The positions of the switches in calibration mode are listed in Table 18-2.

**Table 18-2. Calibration Reference Voltages<sup>† (1)</sup>**

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	$V_{in}$

<sup>(1)</sup> The state of the switches in this table assumes that self-test mode is not enabled.

When CAL ST (ADCALCR.16) is set, a calibration conversion is started. The voltage source selected via the bits BRIDGE EN and HILO is converted once (single conversion mode) and the digital result is returned to the calibration and correction register, ADCALR, where it can be read by the CPU. The CAL ST bit acts as a flag and must be polled by the CPU. It is held set during the conversion process and automatically clears to indicate the end of the reference voltage conversion.

---

**NOTE: No Interrupt for end of calibration**

The ADC does not generate an interrupt to signal the end of the calibration conversion. The application must poll the CAL ST bit to determine the end of the calibration conversion.

---

After the CAL ST bit is set by the application program, it can only be reset by the end of the ongoing conversion generated by the ADC core. If the calibration conversion is interrupted (CAL EN bit is cleared), the CAL ST bit is held at 1 until a new calibration conversion has been set and completed. Setting the CAL ST bit while calibration is disabled (CAL EN = 0) has no effect; however, in this situation, setting CAL EN immediately starts a calibration conversion. When the calibration conversion is interrupted by an ADC enable (ADC EN = 0, CAL EN = 1, and CAL ST = 1), a new conversion is automatically restarted as soon as the ADC enable bit is released (ADC EN = 1).

### 18.8.1.2 Calibration and Offset Error Correction Sequences

The number of measurements and the source to measure for an ADC calibration are application dependent. The CAL ST bit must be set for each calibration source to be measured. While calibration mode is enabled, any available calibration sources can be converted according to the BRIDGE EN and HILO bits (see [Table 18-2](#)). The digital results of the calibration measurements should be read from ADCALR by the application after each reference conversion so that a correction value can be computed and written back into ADCALR.

When the application has the necessary calibration data, it should compute the offset error correction value and load it into the calibration and correction register, ADCALR. After the CAL EN bit is cleared, normal conversion mode restarts, continuing from where it was frozen, but with the addition of self-correction data.

In normal mode, the self-correction system adds the correction value stored in ADCALR to each digital result before it is written to the respective group's FIFO.

The basic calibration routine is as follows:

1. Enable calibration via CAL EN (ADCALCR.0).
2. Select the voltage source via BRIDGE EN and HILO (ADCALCR.9:8).
3. Start the conversion with CAL ST (ADCALCR.16).
4. Wait for CAL ST to go to 0.
5. Get the results from ADCALR and save to memory.
6. Loop to step 2 until the calibration conversion data is collected for the desired reference voltages.
7. Compute the error correction value using calibration data saved in memory.
8. Load the ADCALR register with the 2s complement of the computed error correction value.
9. Disable calibration mode.



At this point, the ADC can be configured for normal operation and it corrects each digital result with the error correction value loaded in ADCALR.

---

**NOTE: Prevent ADC Calibration Data From Being Overwritten**

In calibration mode, the conversion result is written to ADCALR which overwrites any previous calibration data; therefore, the ADCALR register must be read before a new conversion is started.

---

For no correction, a value of 0x0000 must be written to ADCALR. In noncalibration mode, the ADCALR register can be read and written. Any value written to ADCALR in normal mode (CAL EN = 0) is added to each digital result from the ADC core.

### 18.8.1.3 Mid-Point Calibration

Because of its connections to the ADC's reference voltage (VrefHi, VrefLo), the precision of the calibration reference is voltage independent. On the other hand, the accuracy of the switched bridge resistor (R1 & 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%).

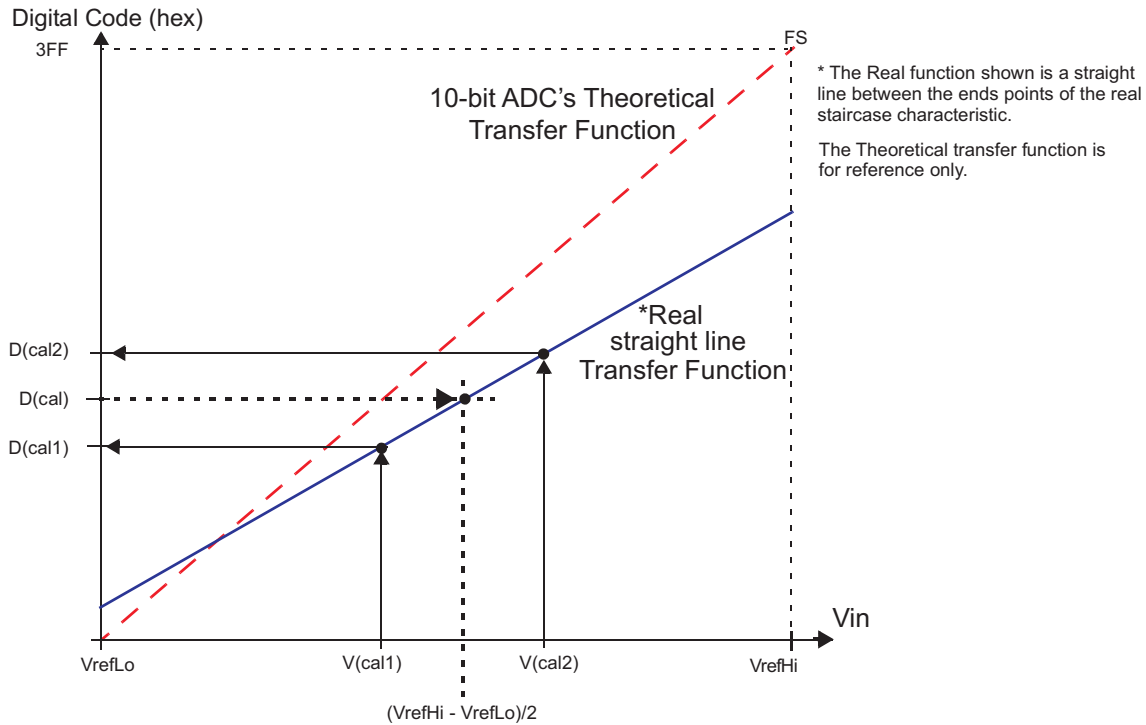
The switched reference voltage device has been specially designed to support a differential measurement of its mid-point voltage. This ensures the accuracy of the mid-point reference, and hence the efficiency of the calibration.

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

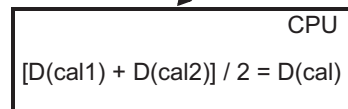
1. The application program connects the voltage VrefHi to R1 and VrefLo to R2, (BRIDGE\_EN=0, HILO=0), launches a conversion of the input voltage V(cal1), and stores the digital result D(cal1) into the memory.
2. Then the application program switches the voltage VrefHi to R2 and VrefLo to R1 (BRIDGE\_EN=0, 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 18-11](#) below summarizes the mid-point calibration flow.

Figure 18-11. Mid-point Value Calculation



$$V(cal1) = [VREFHI * R1 + VREFLO * R2] / (R1 + R2)$$

$$V(cal2) = [VREFLO * R1 + VREFHI * R2] / (R1 + R2)$$

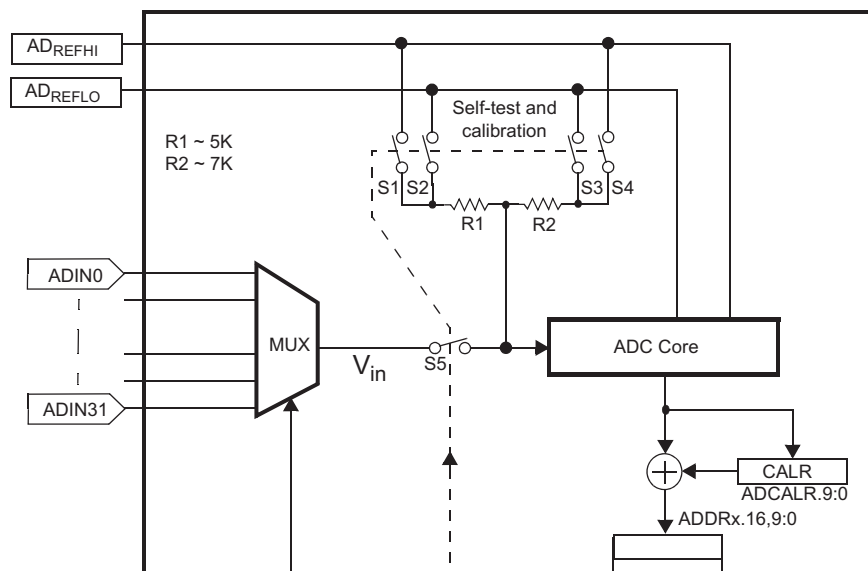


$$[V(cal1) + V(cal2)] / 2 = (VrefHi - VrefLo) / 2$$

### 18.8.2 ADC Self-Test Mode

The ADC module supports a self-test mode which can be used to detect an open or a short on the ADC input channels. Self-test mode is enabled by setting the SELF TST bit (ADCALCR.24). Any conversion type (continuous or single conversion, freeze enabled or non-freeze enabled, interrupts enabled or disabled) can be performed in this mode.

In normal mode, setting the self-test mode while a conversion sequence is in process can corrupt the current channel conversion results. However, the next channel in the sequence is converted correctly during the additional self-test cycle. The logic associated with both self-test and calibration is shown in Figure 18-12.

**Figure 18-12. Self-Test and Calibration Logic**

In self-test mode, a test voltage defined by the HILO bit (ADCALCR.8) is provided to the ADC core input through a resistor (see Table 18-3). To change the test source, this bit can be toggled before any single conversion mode request. Changing this bit while a conversion is in progress *can* corrupt the results if the source switches during the acquisition period.

Please note that the switch S5 shown in the above figure is only for the purpose of explaining the self-test sequence. There is no physical switch.

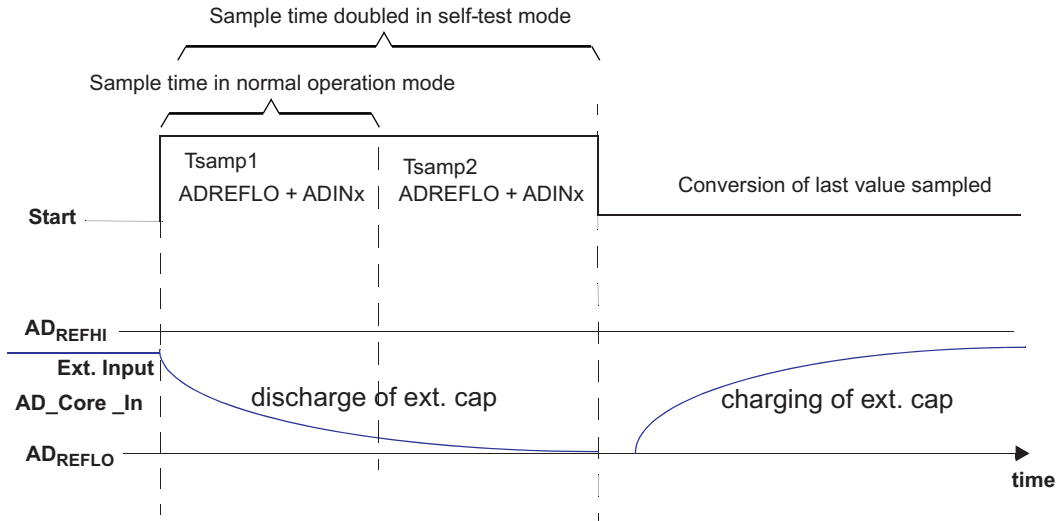
**Table 18-3. Self-Test Reference Voltages<sup>† (1)</sup>**

SELF TST	HILO	S1	S2	S3	S4	S5	Reference Voltage
1	0	0	1	1	0	1	AD <sub>REFLO</sub> via R1    R2 connected to V <sub>in</sub>
1	1	1	0	0	1	1	AD <sub>REFHI</sub> via R1    R2 connected to V <sub>in</sub>
0	X	0	0	0	0	1	V <sub>in</sub>

<sup>(1)</sup> Switches refer to Figure 18-12.

Conversions in self-test mode are started just as they are in the normal operating mode (see Section 18.3.6). The conversion starts according to the configuration set in the three mode control registers (ADEVMODECR, ADG1MODECR, ADG2MODECR) and the sampling time control registers (ADEVSAMP, ADG1SAMP, ADG2SAMP). The acquisition time for each conversion in self-test mode is extended to twice the normal configured acquisition time. The selected reference voltage and the input voltage from the ADIN<sub>x</sub> input channel are both connected to the ADC internal sampling capacitor throughout this extended acquisition period. Figure 18-13 shows the self-test mode timing when the AD<sub>REFLO</sub> is chosen as the reference voltage for the self-test mode conversion. It also assumes an external capacitor connected to the ADC input channel.

**Figure 18-13. Timing for Self-Test Mode**



### 18.8.2.1 Use of Self-Test Mode to Determine Open/Short on ADC Input Channels

The following sequence needs to be used to deduce the ADC pin status:

- Convert the channel with self test enabled and with the reference voltage as Vreflo. Store the conversion result, say Vd.
- Convert the channel with self test enabled and with the reference voltage as Vrefhi. Store the conversion result, say Vu.
- Convert the channel with self test disabled. Store the conversion result, say Vn.

The results can be interpreted using the following table.

**Table 18-4. Determination of ADC Input Channel Condition**

Normal Conversion Result, Vn	Self-test Conversion Result, Vu	Self-test Conversion Result, Vd	Pin Condition
Vn	$Vn < Vu < AD_{REFHI}$	$AD_{REFLO} < Vd < Vn$	Good
$AD_{REFHI}$	$AD_{REFHI}$	approx. $AD_{REFHI}$	Shorted to $AD_{REFHI}$
$AD_{REFLO}$	approx. $AD_{REFLO}$	$AD_{REFLO}$	Shorted to $AD_{REFLO}$
Unknown	$AD_{REFHI}$	$AD_{REFLO}$	Open

### 18.8.3 ADC Power-Down Mode

This is an inactive mode in which the clocks to the ADC module are stopped leaving the module in a static state. The clock to the ADC core (ADCLK) is stopped whenever there are no ongoing conversions. This is the clock-gating implementation requirement. Also, the ADC module places the ADC core into the power down mode such that there is minimal current drawn from the ADC operating and reference supplies.

#### 18.8.3.1 Powering Down Just The ADC Core

The ADC core can be individually powered down without stopping the clocks to the ADC module. This can be done by setting the POWERDOWN bit of the ADC Operating Mode Control Register (ADOPMODECR.3). Whenever a conversion is required the POWERDOWN bit must be cleared, and a minimum time  $t_{d(PU-ADV)}$ , (see the specific device data sheet for actual value) has to be allowed before starting a new conversion. This wait must be implemented in the application software.

### 18.8.3.2 Enhanced Power-Down Mode

A bit in the ADC operating mode control register, IDLE PWRDN (ADOPMODECR.4) enables the enhanced power-down mode of the ADC.

Once this bit is set, the ADC module will power down the ADC core whenever there are no more ongoing or pending ADC conversions. The ADC core will be powered down regardless of the state of the POWERDOWN bit (ADOPMODECR.3).

The ADC module releases the ADC core from power down mode as soon as a new conversion is requested. The ADC logic state machine then has to wait for at least  $t_{d(PU-ADV)}$ , (see the device data sheet for actual value) before starting a new conversion. The IDLE PWRDN bit will remain set at all times. The logic state machine can use this bit to determine that it needs to wait for a programmable number of VCLK cycles before it allows the input channel to be sampled. This time is configured by the ADC Power Up Delay Control register (ADPWRUPDLYCTRL).

If IDLE PWRDN is not set, the ADC module does not wait for any additional delay before sampling the input channel and the application software has to take account of this required delay.

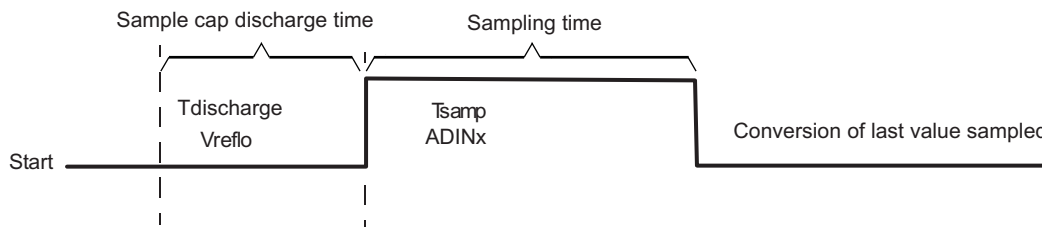
### 18.8.3.3 Managing Clocks to the ADC Module

The clock to the ADC module can be turned off via the appropriate Peripheral Central Resource (PCR) controller PSPWRDNSET register (check the specific device datasheet to identify the register and the bit to be set). If a conversion is ongoing when this bit is set, the ADC module will wait until the current conversion completes before allowing the ADC module clock to be stopped.

### 18.8.4 ADC Sample Capacitor Discharge Mode

This mode allows the charge on the ADC core's internal sampling capacitor to be discharged before starting the sampling phase of the next channel.

**Figure 18-14. 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 18-14. 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.

## 18.9 ADC Results' RAM Special Features

The following sections describe some of the special features supported by the ADC module to enhance the results' RAM testability and integrity.

### 18.9.1 ADC Results' RAM Auto-Initialization

The ADC module allows the application to auto-initialize the ADC results' RAM to all zeros. The application must ensure that the ADC module is not in any of the conversion modes before triggering off the auto-initialization process.

The auto-initialization sequence is as follows:

1. Enable the global hardware memory initialization key by programming a value of 1010 to the bits [3-0] of the MINITGCR register of the System module.
2. Set the control bit for the ADC results' RAM in the MSINENA System module register. The bit 8 of the MSINENA register is used to control the initialization of the ADC1 results' RAM, while bit 14 controls the initialization of the ADC2 results' RAM. This starts the initialization process. The BUF INIT ACTIVE flag in the ADC module ADBNDEND register will get set to reflect that the initialization is ongoing.
3. When the memory initialization is completed, the corresponding status bit in the MINISTAT register will be set. Also, the BUF INIT ACTIVE flag will get cleared.

### 18.9.2 ADC Results' RAM Test Mode

In the defined conversion modes of the ADC, the application can only read from the ADC results' RAM. Only the ADC module is allowed to write to the results' RAM. A special test mode is defined to allow the application to also write into the ADC results' RAM - this mode is the ADC Results' RAM Test Mode. Only 32-bit reads and writes are allowed to the ADC results' RAM in this test mode.

---

**NOTE: Contention on access to ADC Results' RAM**

The ADC module cannot handle a contention between the application write to the results' RAM and the ADC writing a conversion result to the results' RAM. The application must ensure that the ADC is not likely to write a new conversion result to the results' RAM when the ADC Results' RAM Test Mode is enabled.

---

The ADC Results' RAM Test Mode is enabled by setting the RAM TEST EN bit in the ADC OP MODE CR.

### 18.9.3 ADC Results' RAM Parity

The below shows the ADC Results' RAM parity control registers.

Parity checking is implemented using parity on a per-half word basis for the ADC RAM. That is, there is one parity bit for 16 bits of the ADC RAM. The polarity of the ADC RAM parity is controlled by the DEVCR1 register in the system module (address = 0xFFFFFDC). The parity checking is enabled by the ADPARCR register. After reset, the parity checking is disabled and must be enabled if parity protection is required.

During a read access, the parity is calculated based on the data read from the ADC RAM and compared with the good parity value stored in the parity bits. If any word fails the parity check then the ADC generates an error signal hooked up to the Error Signaling Module (ESM). The ADC RAM address which generated the parity error is captured for host system debugging, and is frozen from being updated until it is read by the application.

#### Testing the Parity Checking Mechanism:

To test the parity checking mechanism itself, the parity RAM is made writable by the CPU in a special test mode. This is done by a control bit called TEST in the AD PAR CR register. Once this bit is set, the parity bits are mapped to an address starting at an address offset of 4KB from the base address of the ADC RAM. See [Figure 18-15](#). The CPU can now manually insert parity errors. Note that the ADC RAM only supports 32-bit accesses.



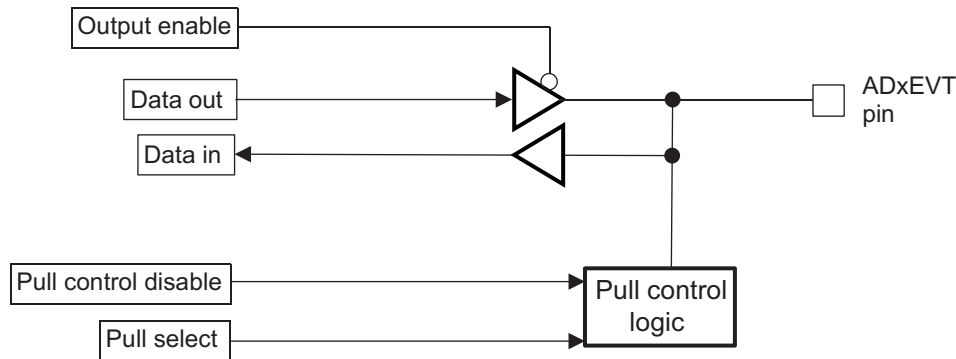
## 18.10 ADEVT Pin General Purpose I/O Functionality

The AD1EVT pin for ADC1 and AD2EVT pin for ADC2 can be configured as general-purpose I/O signals. The following sections describe the different ways in which the application can configure the ADxEVT pins.

### 18.10.1 GPIO Functionality

Figure 18-16 illustrates the GPIO functionality of the ADxEVT pin.

Figure 18-16. GPIO Functionality of ADxEVT



Once the device power-on reset is released, the ADC module controls the state of the ADxEVT pin.

- **Pull control:** The pull control can either be enabled or disabled by default (while system reset is active and after it is released). The actual default state of the pull control is specified in the device datasheet. The application can enable pull control by clearing the PDIS (pull control disable) bit in the ADEVTPDIS register. In this case, if the PSEL (pull select) bit in the ADEVTPSEL register is set, the pin will have a pull-up. If the PSEL bit is cleared, the pin will have a pull-down. If the PDIS bit is set in the control register, there is no pull-up or pull-down on the pin.

---

**NOTE: Pull Behavior when ADxEVT is configured as output**

If the ADxEVT pin is configured as output, then the pulls are disabled automatically. If the pin is configured as input, the pulls are enabled or disabled depending on bit PDIS in the pull disable register ADEVTPDIS.

---

- **Output buffer:** The ADxEVT pin can be driven as an output pin if the ADEVTDIR bit is set in the pin direction control register.
- **Open-Drain Feature:** The open drain output capability is enabled via the ADEVTPDR control register. The ADxEVT pin must be also configured to be an output pin for this mode.
  - The output buffer is enabled if a low signal is being driven on to the pin.
  - The output buffer is disabled if a high signal is being driven on to the pin.

### 18.10.2 Summary

The behavior of the output buffer, and the pull control is summarized in Table 18-5. The input buffer for the ADxEVT pins are enabled once the device power-on reset is released.



**Table 18-5. Output Buffer, and Pull Control Behavior for ADxEVT as GPIO Pins <sup>(1)</sup>**

System Reset Active?	Pin Direction (DIR)	Pull Disable (PDIS)	Pull Select (PSEL)	Pull Control	Output Buffer
Yes	X	X	X	Enabled	Disabled
No	0	0	0	Pull down	Disabled
No	0	0	1	Pull up	Disabled
No	0	1	0	Disabled	Disabled
No	0	1	1	Disabled	Disabled
No	1	X	X	Disabled	Enabled

(1)

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

## 18.11 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 application must ensure that the reserved bits are always written as 0 to ensure software compatibility to future revisions of the module. The table below shows register address offsets from the base address of the ADC modules. The base address of ADC1 registers is FFF7 C000h and the base address of ADC2 registers is FFF7 C200h.

**Table 18-6. ADC Registers Summary**

Offset	Acronym	Register Description	Section
000	ADRSTCR	ADC Reset Control Register	<a href="#">Section 18.11.1</a>
004h	ADOPMODECR	ADC Operating Mode Control Register	<a href="#">Section 18.11.2</a>
008h	ADCLOCKCR	ADC Clock Control Register	<a href="#">Section 18.11.3</a>
00Ch	ADCALCR	ADC Calibration Mode Control Register	<a href="#">Section 18.11.4</a>
010h	ADEVMODECR	ADC Event Group Operating Mode Control Register	<a href="#">Section 18.11.5</a>
014h	ADG1MODECR	ADC Group1 Operating Mode Control Register	<a href="#">Section 18.11.6</a>
018h	ADG2MODECR	ADC Group2 Operating Mode Control Register	<a href="#">Section 18.11.7</a>
01Ch	ADEVSR	ADC Trigger Source Select Register	<a href="#">Section 18.11.8</a>
020h	ADG1SR	ADC Group1 Trigger Source Select Register	<a href="#">Section 18.11.9</a>
024h	ADG2SR	ADC Group2 Trigger Source Select Register	<a href="#">Section 18.11.10</a>
028h	ADEVINTENA	ADC Event Interrupt Enable Control Register	<a href="#">Section 18.11.11</a>
02Ch	ADG1INTENA	ADC Group1 Interrupt Enable Control Register	<a href="#">Section 18.11.12</a>
030h	ADG2INTENA	ADC Group2 Interrupt Enable Control Register	<a href="#">Section 18.11.13</a>
034h	ADEVINTFLG	ADC Event Group Interrupt Flag Register	<a href="#">Section 18.11.14</a>
038h	ADG1INTFLG	ADC Group1 Interrupt Flag Register	<a href="#">Section 18.11.15</a>
03Ch	ADG2INTFLG	ADC Group2 Interrupt Flag Register	<a href="#">Section 18.11.16</a>
040h	ADEVTHRINTCR	ADC Event Group Threshold Interrupt Control Register	<a href="#">Section 18.11.17</a>
044h	ADG1THRINTCR	ADC Group1 Threshold Interrupt Control Register	<a href="#">Section 18.11.18</a>
048h	ADG2THRINTCR	ADC Group2 Threshold Interrupt Control Register	<a href="#">Section 18.11.19</a>
04Ch	ADEVDMACR	ADC Event Group DMA Control Register	<a href="#">Section 18.11.20</a>
050h	ADG1DMACR	ADC Group1 DMA Control Register	<a href="#">Section 18.11.21</a>
054h	ADG2DMACR	ADC Group2 DMA Control Register	<a href="#">Section 18.11.22</a>
058h	ADBNDCR	ADC Results Memory Configuration Register	<a href="#">Section 18.11.23</a>
05Ch	ADBNDEND	ADC Results Memory Size Configuration Register	<a href="#">Section 18.11.24</a>
060h	ADEVSAMP	ADC Event Group Sampling Time Configuration Register	<a href="#">Section 18.11.25</a>
064h	ADG1SAMP	ADC Group1 Sampling Time Configuration Register()	<a href="#">Section 18.11.26</a>
068h	ADG2SAMP	ADC Group2 Sampling Time Configuration Register	<a href="#">Section 18.11.27</a>
06Ch	ADEVSR	ADC Event Group Status Register	<a href="#">Section 18.11.28</a>
070h	ADG1SR	ADC Group1 Status Register	<a href="#">Section 18.11.29</a>
074h	ADG2SR	ADC Group2 Status Register	<a href="#">Section 18.11.30</a>
078h	ADEVSEL	ADC Event Group Channel Select Register	<a href="#">Section 18.11.31</a>
07Ch	ADG1SEL	ADC Group1 Channel Select Register	<a href="#">Section 18.11.32</a>
080h	ADG2SEL	ADC Group2 Channel Select Register	<a href="#">Section 18.11.33</a>
084h	ADCALR	ADC Calibration and Error Offset Correction Register	<a href="#">Section 18.11.34</a>
088h	ADSMSTATE	ADC State Machine Status Register	<a href="#">Section 18.11.35</a>
08Ch	ADLASTCONV	ADC Channel Last Conversion Value Register	<a href="#">Section 18.11.36</a>
090h - 0AFh	ADEVBUFFER	ADC Event Group Results FIFO Register	<a href="#">Section 18.11.37</a>
0B0h - 0CFh	ADG1BUFFER	ADC Group1 Results FIFO Register	<a href="#">Section 18.11.38</a>
0D0h - 0EFh	ADG2BUFFER	ADC Group2 Results FIFO Register	<a href="#">Section 18.11.39</a>

**Table 18-6. ADC Registers Summary (continued)**

Offset	Acronym	Register Description	Section
0F0h	ADEVEMUBUFFER	ADC Event Group Results Emulation FIFO Register	<a href="#">Section 18.11.40</a>
0F4h	ADG1EMUBUFFER	ADC Group1 Results Emulation FIFO Register	<a href="#">Section 18.11.41</a>
0F8h	ADG2EMUBUFFER	ADC Group2 Results Emulation FIFO Register	<a href="#">Section 18.11.42</a>
0FCh	ADEVTDIR	ADC ADEVT Pin Direction Control Register	<a href="#">Section 18.11.43</a>
100h	ADEVTOUR	ADC ADEVT Pin Output Value Control Register	<a href="#">Section 18.11.44</a>
104h	ADEVTIN	ADC ADEVT Pin Input Value Register	<a href="#">Section 18.11.45</a>
108h	ADEVTSER	ADC ADEVT Pin Set Register	<a href="#">Section 18.11.46</a>
10Ch	ADEVTCLR	ADC ADEVT Pin Clear Register	<a href="#">Section 18.11.47</a>
110h	ADEVTPDR	ADC ADEVT Pin Open Drain Enable Register	<a href="#">Section 18.11.48</a>
114h	ADEVTPDIS	ADC ADEVT Pin Pull Control Disable Register	<a href="#">Section 18.11.49</a>
118h	ADEVTPSEL	ADC ADEVT Pin Pull Control Select Register	<a href="#">Section 18.11.50</a>
11Ch	ADEVSAMPDISEN	ADC Event Group Sample Cap Discharge Control Register	<a href="#">Section 18.11.51</a>
120h	ADG1SAMPDISEN	ADC Group1 Sample Cap Discharge Control Register	<a href="#">Section 18.11.52</a>
124h	ADG2SAMPDISEN	ADC Group2 Sample Cap Discharge Control Register	<a href="#">Section 18.11.53</a>
128h - 138h	ADMAGINTxCR	ADC Magnitude Compare Interrupt Control Register	<a href="#">Section 18.11.54</a>
12Ch - 13Ch	ADMAGxMASK	ADC Magnitude Compare Mask Register	<a href="#">Section 18.11.55</a>
158h	ADMAGINTENASET	ADC Magnitude Compare Interrupt Enable Set Register	<a href="#">Section 18.11.56</a>
15Ch	ADMAGINTENACLAR	ADC Magnitude Compare Interrupt Enable Clear Register	<a href="#">Section 18.11.57</a>
160h	ADMAGINTFLG	ADC Magnitude Compare Interrupt Flag Register	<a href="#">Section 18.11.58</a>
164h	ADMAGINTOFF	ADC Magnitude Compare Interrupt Offset Register	<a href="#">Section 18.11.59</a>
168h	ADEVFIFORESETCR	ADC Event Group FIFO Reset Control Register	<a href="#">Section 18.11.60</a>
16Ch	ADG1FIFORESETCR	ADC Group1 FIFO Reset Control Register	<a href="#">Section 18.11.61</a>
170h	ADG2FIFORESETCR	ADC Group2 FIFO Reset Control Register	<a href="#">Section 18.11.62</a>
174h	ADEVRAMWRADDR	ADC Event Group RAM Write Address Register	<a href="#">Section 18.11.63</a>
178h	ADG1RAMWRADDR	ADC Group1 RAM Write Address Register	<a href="#">Section 18.11.64</a>
17Ch	ADG2RAMWRADDR	ADC Group2 RAM Write Address Register	<a href="#">Section 18.11.65</a>
180h	ADPARCR	ADC Parity Control Register	<a href="#">Section 18.11.66</a>
184h	ADPARADDR	ADC Parity Error Address Register	<a href="#">Section 18.11.67</a>
188h	ADPWRUPDLYCTRL	ADC Power-Up Delay Control Register	<a href="#">Section 18.11.68</a>

### 18.11.1 ADC Reset Control Register (ADRSTCR)

Figure 18-17 and Table 18-7 describe the ADRSTCR register.

**Figure 18-17. ADC Reset Control Register (ADRSTCR)**

31	Reserved	1	0
R-0		RESET R/W-0	

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

**Table 18-7. ADC Reset Control Register (ADRSTCR) Field Descriptions**

Bit	Field	Value	Description
31-1	Reserved	0	Reads return zeros, writes have no effect.
0	RESET	0	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. Module is released from the reset state.
		1	All the module's internal state machines and the control/status registers are reset.

### 18.11.2 ADC Operating Mode Control Register (ADOPMODECR)

Figure 18-18 and Table 18-8 describe the ADOPMODECR register.

**Figure 18-18. ADC Operating Mode Control Register (ADOPMODECR)**

31	30	25	24	23	21	20	17	16			
10/12 BIT	Reserved			COS	Reserved		CHN TEST EN	RAM TEST EN			
R/W-0	R-0			R/W-0	R-0		R/W-Ah	R/W-0			
15	Reserved			9	8	7	5	4	3	1	0
Reserved				POWE R DOWN	Reserved		IDLE PWRD N	Reserved		ADC EN	
R-0				R/W-0	R-0		R/W-0	R-0		R/W-0	

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

**Table 18-8. ADC Operating Mode Control Register (ADOPMODECR) Field Descriptions**

Bit	Field	Value	Description
31	10/12 BIT	0 1	This bit controls the resolution of the ADC core. It also affects the size of the conversion results stored in the results' RAM. Any operation mode read/write: 0 The ADC core and digital logic are configured to be in 10-bit resolution. This is the default mode of operation. 1 The ADC core and digital logic are configured to be in 12-bit resolution.
30-25	Reserved	0	Read returns 0. Writes have no effect.
24	COS	0 1	This bit affects <i>emulation operation only</i> . It defines whether the ADC core clock (ADCLK) is immediately halted when the emulation system enters suspend mode or if it should continue operating normally. <b>Note:</b> 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. Any operation 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	Read returns 0. Writes have no effect.
20-17	CHN TEST EN	Ah 5h other	Enable the input channels' impedance measurement mode. <b>This mode is reserved for use by TI.</b> Any operation mode read/write: Ah Input impedance measurement mode is disabled. 5h Input impedance measurement mode is enabled. other Input impedance measurement mode is disabled.
16	RAM TEST EN	0 1	Enable the ADC Results' RAM Test Mode. Please refer to <a href="#">Section 18.9.2</a> for more details. Any operation mode read/write: 0 ADC RAM Test Mode is disabled. The application cannot write to the ADC RAM by the CPU or the DMA. 1 ADC RAM Test Mode is enabled. The application can directly write to the ADC RAM by the CPU or the DMA.
15-9	Reserved	0	Read returns 0. Writes have no effect.

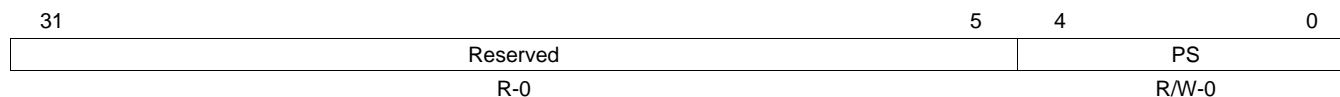
**Table 18-8. ADC Operating Mode Control Register (ADOPMODECR) Field Descriptions (continued)**

Bit	Field	Value	Description
8	POWER DOWN		ADC Power Down. This bit powers down only the ADC core; the digital logic in the sequencer stays active. To release the core from power down mode, this bit must be cleared. If a conversion is ongoing, the ADC module will wait until the current conversion is completed before powering down the ADC core.  Also refer <a href="#">Section 18.11.68</a> , ADC Power-Up Delay Control Register (ADPWRUPDLYCTRL). Any operation mode read/write:
		0	The state of the ADC core is controlled by the IDLE PWRDN bit, or by a global power down mode entry.
		1	ADC core is in the power-down state.
7-5	Reserved	0	Read returns 0. Writes have no effect.
4	IDLE PWRDN		ADC Power Down When Idle. When this bit is set, the ADC module will automatically power down the ADC core whenever there are no conversions ongoing or pending. This is the enhanced power down mode.  Also refer <a href="#">Section 18.11.68</a> , ADC Power-Up Delay Control Register (ADPWRUPDLYCTRL). Any operation mode read/write:
		0	The ADC stays in the normal operating mode even if no conversions are ongoing or pending. The power down state is entered only by configuring the POWER DOWN bit or via a global power down mode entry.
		1	Enhanced power down mode is enabled.
3-1	Reserved	0	Read returns 0. 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.  Any operation 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.

### 18.11.3 ADC Clock Control Register (ADCLOCKCR)

Figure 18-19 and Table 18-9 describe the ADCLOCKCR register.

**Figure 18-19. ADC Clock Control Register (ADCLOCKCR)**



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

**Table 18-9. ADC Clock Control Register (ADCLOCKCR) Field Descriptions**

Bit	Field	Value	Description
31-5	Reserved	0	Reads return zeros, writes have no effect.
4-0	PS	0-1Fh	<p>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.</p> <p><b>Note:</b> The supported range for the ADC clock frequency is specified in the device datasheet. The ADC clock prescaler must be configured to meet this datasheet specification.</p> <p>Any operation mode read/write:</p> $t_{C(ADCLK)} = t_{C(VCLK)} * (PS[4:0] + 1),$ <p>where <math>t_{C(ADCLK)}</math> is the period of the ADCLK, and <math>t_{C(VCLK)}</math> is the period of the VCLK.</p>

### 18.11.4 ADC Calibration Mode Control Register (ADCALCR)

Figure 18-20 and Table 18-10 describe the ADCALCR register.

**Figure 18-20. ADC Calibration Mode Control Register (ADCALCR)**

31	25	24	23	17	16
Reserved		SELF TEST	Reserved		CAL ST
R-0		R/W-0	R-0		R/WS- 0
15	10	9	8	7	1
Reserved		BRIDG E EN	HILO	Reserved	
R-0		R/W-0	R/W-0	R-0	
				CAL EN	
				R/W-0	

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

**Table 18-10. ADC Calibration Mode Control Register (ADCALCR) Field Descriptions**

Bit	Field	Value	Description
31-25	Reserved	0	Read returns 0. Writes have no effect.
24	SELF TEST	0 1	ADC Self Test Enable. When this bit is Set, either AD <sub>REFHI</sub> or AD <sub>REFLO</sub> is connected through a resistor to the selected input channel. The desired conversion mode is configured in the group mode control registers. For more details on the ADC Self Test Mode, please refer to <a href="#">Section 18.8.2</a> .  Any operation mode read/write: 0 ADC Self Test mode is disabled. 1 ADC Self Test mode is enabled.
23-17	Reserved	0	Read returns 0. Writes have no effect.
16	CAL ST	0 1	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: 0 Read: Calibration conversion is in progress. Write: ADC module starts calibration conversion. 1 Read: Calibration conversion has completed, or has not yet been started. Write: Writing 0 to this bit has no effect.
15-10	Reserved	0	Read returns 0. Writes have no effect.
9	BRIDGE EN		Bridge Enable. When set with the HILO bit, BRIDGE EN allows a reference voltage to be converted in calibration mode. <a href="#">Table 18-2</a> defines the four different reference voltages that can be selected.
8	HILO		ADC Self Test mode and Calibration Mode Reference Source Selection.  In the ADC Self Test mode, this bit defines the test voltage to be combined through a resistor with the selected input pin voltage. Refer <a href="#">Section 18.8.2</a> for details on the ADC Self Test Mode.  In the ADC Calibration Mode, this bit defines the reference source polarity. Refer <a href="#">Section 18.8.1</a> for details on the ADC Calibration Mode.  In the ADC module's normal operating mode, this bit has no effect.
7-1	Reserved	0	Read returns 0. Writes have no effect.
0	CAL EN	0 1	ADC Calibration Enable. When this bit is Set, the input channel multiplexor is disconnected and the calibration reference voltage is connected to the ADC core input. The calibration reference voltage is selected by the combination of the BRIDGE EN and HILO. The actual conversion of this reference voltage starts when the CAL ST bit is set. If the CAL ST bit is already set when the CAL EN bit is set, then the calibration conversion is immediately started.  Please refer to <a href="#">Section 18.8.1</a> for more details on the ADC calibration mode.  Any operation mode read/write: 0 Calibration mode is disabled. 1 Calibration mode is enabled.



### 18.11.5 ADC Event Group Operating Mode Control Register (ADEVMODECR)

ADC Event Group Operating Mode Control Register (ADEVMODECR) is shown in [Figure 18-21](#) and described in [Table 18-11](#).

**Figure 18-21. ADC Event Group Operating Mode Control Register (ADEVMODECR)**

31	Reserved						24
R-0							
23	Reserved				17	16	
R-0					No Reset on ChnSel		
R/W-0							
15	Reserved			10	9	8	
R-0				EV DATA FMT			
R/W-0							
7	6	5	4	3	2	1	0
Reserved		EV CHID	OVR EV RAM IGN	Reserved	EV 8BIT	EV MODE	FRZ EV
R-0		R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0

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

**Table 18-11. ADC Event Group Operating Mode Control Register (ADEVMODECR) Field Descriptions**

Bit	Field	Value	Description
31-17	Reserved	0	Reads return zeros, writes have no effect.
16	No Reset on ChnSel	0 1	<p>No Event Group Results Memory Reset on New Channel Select.</p> <p>This bit determines whether the event group results' RAM is reset whenever a non-zero value is written to the event group channel select register.</p> <p>Any operation mode read/write:</p> <p>0 Event group results RAM is reset when a non-zero value is written to event group channel select register, even if event group conversions are completed.</p> <p>1 Event group results RAM is not reset when a non-zero value is written to event group channel select register, and event group conversions are completed.</p> <p>If the event group conversions are ongoing (active or frozen), then writing a non-zero value to the event group channel select register will always reset the event group results RAM.</p>
15-10	Reserved	0	Reads return zeros, writes have no effect.
9-8	EV DATA FMT	0 1h 2h 3h	<p>Event Group Read Data Format.</p> <p>This field is only applicable when the ADC module is configured to be in the 12-bit resolution mode. This field is reserved when the module is configured as a 10-bit ADC module.</p> <p>This field determines the format in which the conversion results are read out of the Event group results RAM when using the FIFO interface, that is, when reading from the AD EV BUFFER or AD EV EMU BUFFER locations.</p> <p>Any operation mode read/write:</p> <p>0 Conversion results are read out in full 12-bit format. This is the default mode.</p> <p>1h Conversion results are read out in 10-bit format. Bits 11–2 of the 12-bit conversion result are returned as the 10-bit conversion result.</p> <p>2h Conversion results are read out in 8-bit format. Bits 11–4 of the 12-bit conversion result are returned as the 8-bit conversion result.</p> <p>3h Reserved. The full 12-bit conversion result is returned if programmed.</p>
7-6	Reserved	0	Reads return zeros, writes have no effect.

**Table 18-11. ADC Event Group Operating Mode Control Register (ADEVMODECR) Field Descriptions  
(continued)**

Bit	Field	Value	Description
5	EV CHID	<p>0</p> <p>1</p>	<p>Enable Channel Id for the Event Group conversion results to be read. This bit only affects the “read from FIFO” mode. The ADC always stores the channel id in the results RAM. Any 16-bit read performed in the “read from RAM” mode will return the 5-bit channel id along with the 10-bit conversion result.</p> <p>Any operation mode read/write:</p> <p>Bits 14-10, the channel id field, of the data read from the Event Group results' FIFO is read as 00000b.</p> <p>Bits 14-10, the channel id field, of the data read from the Event Group results' FIFO contains the number of the ADC analog input to which the conversion result belongs.</p>
4	OVR EV RAM IGN	<p>0</p> <p>1</p>	<p>This bit allows the ADC module to overwrite the contents of the Event Group results memory under an overrun condition.</p> <p>Any operation mode read/write:</p> <p>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.</p> <p>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.</p>
3	Reserved	0	Reads return zeros, writes have no effect.
2	EV SBIT	<p>0</p> <p>1</p>	<p>Event Group 8-bit result mode. This bit allows the Event Group conversion results to be read out in an 8-bit format. This bit only applies to the “read from FIFO” mode. The lower 2 bits of the 10-bit conversion result are discarded and the upper 8 bits are shifted right two places to form the 8-bit conversion result.</p> <p>Any operation mode read/write:</p> <p>0 The Event Group conversion result is read out as a 10-bit value in the “read from Event Group FIFO” mode.</p> <p>1 The Event Group conversion result is read out as an 8-bit value in the “read from Event Group FIFO” mode.</p>
1	EV MODE	<p>0</p> <p>1</p>	<p>Event Group Conversion Mode. This bit defines whether the input channels selected for conversion in the Event Group are converted only once per trigger, or are continuously converted.</p> <p>Any operation mode read/write:</p> <p>0 The channels selected for conversion in the Event Group are converted only once when the selected event trigger condition occurs.</p> <p>1 The channels selected for conversion in the Event Group are converted continuously when the selected event trigger condition occurs.</p>
0	FRZ EV	<p>0</p> <p>1</p>	<p>Event Group Freeze Enable. This bit allows an Event Group conversion sequence to be frozen if a Group1 or a Group2 conversion is requested. The Event Group conversion is kept frozen until the Group1 or Group2 conversion is active, and continues from where it was frozen once the Group1 or Group2 conversions are completed.</p> <p>While the Event Group conversion is frozen, the EV STOP status flag in the ADEVST register indicates that the Event Group conversions have stopped. This bit gets cleared when the Event Group conversions resume.</p> <p>Any operation mode read/write:</p> <p>0 Event Group conversions cannot be frozen. All the channels selected for conversion in the Event Group are converted before the ADC can switch over to servicing any other conversion group.</p> <p>1 Event Group conversions are frozen whenever there is a request for conversion from Group1 or Group2.</p>

### 18.11.6 ADC Group1 Operating Mode Control Register (ADG1MODECR)

ADC Group1 Operating Mode Control Register (ADG1MODECR) is shown in Figure 18-22 and described in Table 18-12.

**Figure 18-22. ADC Group1 Operating Mode Control Register (ADG1MODECR)**

31	Reserved						24
R-0							
23	Reserved				17	16	
R-0					No Reset on ChnSel		
R/W-0							
15	Reserved			10	9	8	
R-0				G1 DATA FMT			
R/W-0							
7	6	5	4	3	2	1	0
Reserved		G1 CHID	OVR G1 RAM IGN	G1 HW TRIG	G1 8BIT	G1 MODE	FRZ G1
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

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

**Table 18-12. ADC Group1 Operating Mode Control Register (ADG1MODECR) Field Descriptions**

Bit	Field	Value	Description
31-17	Reserved	0	Reads return zeros, writes have no effect.
16	No Reset on ChnSel	0 1	<p>No Group1 Results Memory Reset on New Channel Select.</p> <p>This bit determines whether the group1 results' RAM is reset whenever a non-zero value is written to the group1 channel select register.</p> <p>Any operation mode read/write:</p> <p>0 Group1 results RAM is reset when a non-zero value is written to group1 channel select register, even if group1 conversions are completed.</p> <p>1 Group1 results RAM is not reset when a non-zero value is written to group1 channel select register, and group1 conversions are completed.</p> <p>If the group1 conversions are ongoing (active or frozen), then writing a nonzero value to the group1 channel select register will always reset the group1 results RAM.</p>
15-10	Reserved	0	Reads return zeros, writes have no effect.
9-8	G1 DATA FMT	0 1h 2h 3h	<p>Group1 Read Data Format.</p> <p>This field is only applicable when the ADC module is configured to be in the 12-bit resolution mode. This field is reserved when the module is configured as a 10-bit ADC module.</p> <p>This field determines the format in which the conversion results are read out of the group1 results RAM when using the FIFO interface, that is, when reading from the AD G1 BUFFER or AD G1 EMU BUFFER locations.</p> <p>Any operation mode read/write:</p> <p>0 Conversion results are read out in full 12-bit format. This is the default mode.</p> <p>1h Conversion results are read out in 10-bit format. Bits 11–2 of the 12-bit conversion result are returned as the 10-bit conversion result.</p> <p>2h Conversion results are read out in 8-bit format. Bits 11–4 of the 12-bit conversion result are returned as the 8-bit conversion result.</p> <p>3h Reserved. The full 12-bit conversion result is returned if programmed.</p>
7-6	Reserved	0	Reads return zeros, writes have no effect.

**Table 18-12. ADC Group1 Operating Mode Control Register (ADG1MODECR) Field Descriptions  
(continued)**

Bit	Field	Value	Description
5	G1 CHID	<p>0 Bits 14-10, the channel id field, of the data read from the Group1 results' FIFO is read as 00000b.</p> <p>1 Bits 14-10, the channel id field, of the data read from the Group1 results' FIFO contains the number of the ADC analog input to which the conversion result belongs.</p>	<p>Enable Channel Id for the Group1 conversion results to be read. This bit only affects the "read from FIFO" mode. The ADC always stores the channel id in the results RAM. Any 16-bit read performed in the "read from RAM" mode will return the 5-bit channel id along with the 10-bit conversion result.</p> <p>Any operation mode read/write:</p>
4	OVR G1 RAM IGN	<p>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.</p> <p>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.</p>	<p>This bit allows the ADC module to overwrite the contents of the Group1 results memory under an overrun condition.</p> <p>Any operation mode read/write:</p>
3	G1 HW TRIG	<p>0 The Group1 is software-triggered. A Group1 conversion starts whenever the Group1 channel select register (ADG1SEL) is written with a non-zero value.</p> <p>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).</p>	<p>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 <a href="#">Section 18.3.6</a>.</p> <p>Any operation mode read/write:</p>
2	G1 SBIT	<p>0 The Group1 conversion result is read out as a 10-bit value in the "read from Group1 FIFO" mode.</p> <p>1 The Group1 conversion result is read out as an 8-bit value in the "read from Group1 FIFO" mode.</p>	<p>Group1 8-bit result mode.</p> <p>This field is only applicable when the ADC module is configured to be in the 10-bit resolution mode. This field is reserved when the module is configured as a 12-bit ADC module.</p> <p>This bit allows the Group1 conversion results to be read out in an 8-bit format. This bit only applies to the "read from FIFO" mode. The lower 2 bits of the 10-bit conversion result are discarded and the upper 8 bits are shifted right two places to form the 8-bit conversion result.</p> <p>Any operation mode read/write:</p>
1	G1 MODE	<p>0 The channels selected for conversion in the Group1 are converted only once.</p> <p>1 The channels selected for conversion in the Group1 are converted continuously.</p>	<p>Group1 Conversion Mode. This bit defines whether the input channels selected for conversion in the Group1 are converted only once, or are continuously converted.</p> <p>Any operation mode read/write:</p>
0	FRZ G1	<p>0 Group1 conversions cannot be frozen. All the channels selected for conversion in the Group1 are converted before the ADC can switch over to servicing any other conversion group.</p> <p>1 Group1 conversions are frozen whenever there is a request for conversion from Event Group or Group2.</p>	<p>Group1 Freeze Enable. This bit allows a Group1 conversion sequence to be frozen if an Event Group or a Group2 conversion is requested. The Group1 conversion is kept frozen until the Event Group or Group2 conversion is active, and continues from where it was frozen once the Event Group or Group2 conversions are completed.</p> <p>While the Group1 conversion is frozen, the G1 STOP status flag in the ADG1ST register indicates that the Group1 conversions have stopped. This bit gets cleared when the Group1 conversions resume.</p> <p>Any operation mode read/write:</p>

### 18.11.7 ADC Group2 Operating Mode Control Register (ADG2MODECR)

ADC Group2 Operating Mode Control Register (ADG2MODECR) is shown in Figure 18-23 and described in Table 18-13.

**Figure 18-23. ADC Group2 Operating Mode Control Register (ADG2MODECR)**

31	Reserved						24
R-0							
23	Reserved					No Reset on ChnSel	
R-0						R/W-0	
15	Reserved				10	9	8
R-0					G2 DATA FMT		
R-0					R/W-0		
7	6	5	4	3	2	1	0
Reserved		G2 CHID	OVR G2 RAM IGN	G2 HW TRIG	G2 SBIT	G2 MODE	FRZ EV
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

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

**Table 18-13. ADC Group 2 Operating Mode Control Register (ADG2MODECR) Field Descriptions**

Bit	Field	Value	Description
31-17	Reserved	0	Reads return zeros, writes have no effect.
16	No Reset on ChnSel	0 1	<p>No Group2 Results Memory Reset on New Channel Select.</p> <p>This bit determines whether the group2 results' RAM is reset whenever a non-zero value is written to the group2 channel select register.</p> <p>Any operation mode read/write:</p> <p>0 Group2 results RAM is reset when a non-zero value is written to group2 channel select register, even if group2 conversions are completed.</p> <p>1 Group2 results RAM is not reset when a non-zero value is written to group2 channel select register, and group2 conversions are completed.</p> <p>If the group2 conversions are ongoing (active or frozen), then writing a nonzero value to the group2 channel select register will always reset the group2 results RAM.</p>
15-10	Reserved	0	Reads return zeros, writes have no effect.
9-8	G2 DATA FMT	0 1h 2h 3h	<p>Group2 Read Data Format.</p> <p>This field is only applicable when the ADC module is configured to be in the 12-bit resolution mode. This field is reserved when the module is configured as a 10-bit ADC module.</p> <p>This field determines the format in which the conversion results are read out of the group1 results RAM when using the FIFO interface, that is, when reading from the AD G2 BUFFER or AD G2 EMU BUFFER locations.</p> <p>Any operation mode read/write:</p> <p>0 Conversion results are read out in full 12-bit format. This is the default mode.</p> <p>1h Conversion results are read out in 10-bit format. Bits 11–2 of the 12-bit conversion result are returned as the 10-bit conversion result.</p> <p>2h Conversion results are read out in 8-bit format. Bits 11–4 of the 12-bit conversion result are returned as the 8-bit conversion result.</p> <p>3h Reserved. The full 12-bit conversion result is returned if programmed.</p>
7-6	Reserved	0	Reads return zeros, writes have no effect.

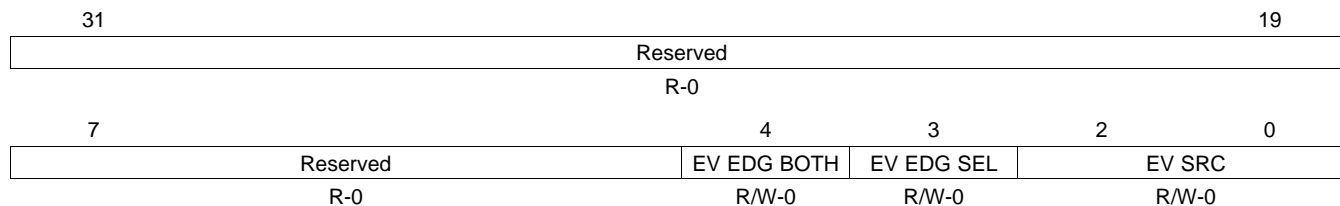
**Table 18-13. ADC Group 2 Operating Mode Control Register (ADG2MODECR) Field Descriptions  
(continued)**

Bit	Field	Value	Description
5	G2 CHID	<p>0</p> <p>1</p>	<p>Enable Channel Id for the Group2 conversion results to be read. This bit only affects the “read from FIFO” mode. The ADC always stores the channel id in the results RAM. Any 16-bit read performed in the “read from RAM” mode will return the 5-bit channel id along with the 10-bit conversion result.</p> <p>Any operation mode read/write:</p> <p>0 Bits 14-10, the channel id field, of the data read from the Group2 results' FIFO is read as 00000b.</p> <p>1 Bits 14-10, the channel id field, of the data read from the Group2 results' FIFO contains the number of the ADC analog input to which the conversion result belongs.</p>
4	OVR G2 RAM IGN	<p>0</p> <p>1</p>	<p>This bit allows the ADC module to overwrite the contents of the Group2 results memory under an overrun condition.</p> <p>Any operation mode read/write:</p> <p>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.</p> <p>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.</p>
3	G2 HW TRIG	<p>0</p> <p>1</p>	<p>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 <a href="#">Section 18.3.6</a>.</p> <p>Any operation mode read/write:</p> <p>0 The Group2 is software-triggered. A Group2 conversion starts whenever the Group2 channel select register (ADG2SEL) is written with a non-zero value.</p> <p>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).</p>
2	G2 SBIT	<p>0</p> <p>1</p>	<p>Group2 8-bit result mode.</p> <p>This field is only applicable when the ADC module is configured to be in the 10-bit resolution mode. This field is reserved when the module is configured as a 12-bit ADC module.</p> <p>This bit allows the Group2 conversion results to be read out in an 8-bit format. This bit only applies to the “read from FIFO” mode. The lower 2 bits of the 10-bit conversion result are discarded and the upper 8 bits are shifted right two places to form the 8-bit conversion result.</p> <p>Any operation mode read/write:</p> <p>0 The Group2 conversion result is read out as a 10-bit value in the “read from Group2 FIFO” mode.</p> <p>1 The Group2 conversion result is read out as an 8-bit value in the “read from Group2 FIFO” mode.</p>
1	G2 MODE	<p>0</p> <p>1</p>	<p>Group2 Conversion Mode. This bit defines whether the input channels selected for conversion in the Group2 are converted only once, or are continuously converted.</p> <p>Any operation mode read/write:</p> <p>0 The channels selected for conversion in the Group2 are converted only once.</p> <p>1 The channels selected for conversion in the Group2 are converted continuously.</p>
0	FRZ G2	<p>0</p> <p>1</p>	<p>Group2 Freeze Enable. This bit allows a Group2 conversion sequence to be frozen if an Event Group or a Group1 conversion is requested. The Group2 conversion is kept frozen until the Event Group or Group1 conversion is active, and continues from where it was frozen once the Event Group or Group1 conversions are completed.</p> <p>While the Group2 conversion is frozen, the G2 STOP status flag in the ADG2ST register indicates that the Group2 conversions have stopped. This bit gets cleared when the Group2 conversions resume.</p> <p>Any operation mode read/write:</p> <p>0 Group2 conversions cannot be frozen. All the channels selected for conversion in the Group2 are converted before the ADC can switch over to servicing any other conversion group.</p> <p>1 Group2 conversions are frozen whenever there is a request for conversion from Event Group or Group1.</p>

### 18.11.8 ADC Event Group Trigger Source Select Register (ADEVSRRC)

ADC Event Group Trigger Source Select Register (ADEVSRRC) is shown in [Figure 18-24](#) and described in [Table 18-14](#).

**Figure 18-24. ADC Event Group Trigger Source Select Register (ADEVSRRC)**



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

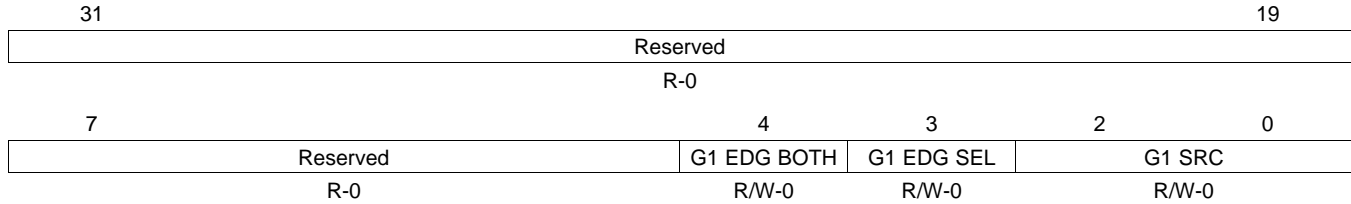
**Table 18-14. ADC Event Group Trigger Source Select Register (ADEVSRRC) Field Descriptions**

Bit	Field	Value	Description
31-5	Reserved	0	Reads return zeros, writes have no effect.
4	EV EDG BOTH	0 1	EV Group Trigger Edge Polarity Select. This bit configures the event group to be triggered on both rising and falling edge detected on the selected trigger source. Any operation mode read/write: 0 The conversion is triggered only upon detecting an edge defined by the EV EDGE SEL bit. 1 The conversion is triggered upon detecting either a rising or falling edge.
3	EV EDG SEL	0 1	Event Group Trigger Edge Polarity Select. This bit determines the polarity of the transition on the selected source that triggers the Event Group conversion. Any operation mode read/write: 0 A high-to-low transition on the selected source will trigger the Event Group conversion. 1 A low-to-high transition on the selected source will trigger the Event Group conversion.
2-0	EV SRC	0-7h	Event Group Trigger Source. Any operation mode read/write: The ADC module allows a trigger source to be selected for the Event Group from up to eight options. These options are device-specific and the device specification must be referred to identify the actual trigger sources.

### 18.11.9 ADC Group1 Trigger Source Select Register (ADG1SRC)

ADC Group1 Trigger Source Select Register (ADG1SRC) is shown in [Figure 18-25](#) and described in [Table 18-15](#).

**Figure 18-25. ADC Group1 Trigger Source Select Register (ADG1SRC)**



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

**Table 18-15. ADC Group1 Trigger Source Select Register (ADG1SRC) Field Descriptions**

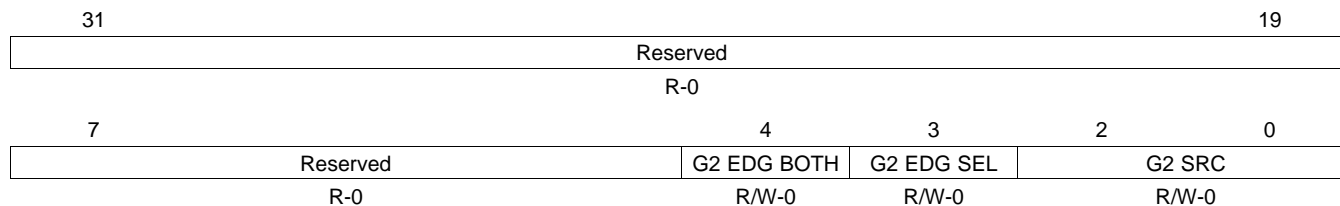
Bit	Field	Value	Description
31-5	Reserved	0	Reads return zeros, writes have no effect.
4	G1 EDG BOTH	0 1	Group1 Trigger Edge Polarity Select. This bit configures the group1 to be triggered on both rising and falling edge detected on the selected trigger source. Any operation mode read/write: 0 The conversion is triggered only upon detecting an edge defined by the G1 EDGE SEL bit. 1 The conversion is triggered upon detecting either a rising or falling edge.
3	G1 EDG SEL	0 1	Group1 Trigger Edge Polarity Select. This bit determines the polarity of the transition on the selected source that triggers the Group1 conversion. Any operation mode read/write: 0 A high-to-low transition on the selected source will trigger the Group1 conversion. 1 A low-to-high transition on the selected source will trigger the Group1 conversion.
2-0	G1 SRC	0-7h	Group1 Trigger Source. Any operation mode read/write: The ADC module allows a trigger source to be selected for the Group1 from up to eight options. These options are device-specific and the device specification must be referred to identify the actual trigger sources.



### 18.11.10 ADC Group2 Trigger Source Select Register (ADG2SRC)

ADC Group2 Trigger Source Select Register (ADG2SRC) is shown in [Figure 18-26](#) and described in [Table 18-16](#).

**Figure 18-26. ADC Group2 Trigger Source Select Register (ADG2SRC)**



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

**Table 18-16. ADC Group2 Trigger Source Select Register (ADG2SRC) Field Descriptions**

Bit	Field	Value	Description
31-5	Reserved	0	Reads return zeros, writes have no effect.
4	G2 EDG BOTH	0 1	Group2 Trigger Edge Polarity Select. This bit configures the group2 to be triggered on both rising and falling edge detected on the selected trigger source. Any operation mode read/write: 0 The conversion is triggered only upon detecting an edge defined by the G2 EDGE SEL bit. 1 The conversion is triggered upon detecting either a rising or falling edge.
3	G2 EDG SEL	0 1	Group2 Trigger Edge Polarity Select. This bit determines the polarity of the transition on the selected source that triggers the Group2 conversion. Any operation mode read/write: 0 A high-to-low transition on the selected source will trigger the Group2 conversion. 1 A low-to-high transition on the selected source will trigger the Group2 conversion.
2-0	G2 SRC	0-7h	Group2 Trigger Source. Any operation mode read/write: The ADC module allows a trigger source to be selected for the Group2 from up to eight options. These options are device-specific and the device specification must be referred to identify the actual trigger sources.

**18.11.11 ADC Event Interrupt Enable Control Register (ADEVINTENA)**

ADC Event Group Interrupt Enable Control Register (ADEVINTENA) is shown in [Figure 18-27](#) and described in [Table 18-17](#).

**Figure 18-27. ADC Event Group Interrupt Enable Control Register (ADEVINTENA)**

31	Reserved					16
	R-0					
15	Reserved					8
	R-0					
7		4	3	2	1	0
	Reserved	EV END INT EN	Reserved	EV OVR INT EN	EV THR INT EN	
	R-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0

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

**Table 18-17. ADC Event Group Interrupt Enable Control Register (ADEVINTENA) Field Descriptions**

Bit	Field	Value	Description
31-4	Reserved	0	Reads return zeros, writes have no effect.
3	EV END INT EN	0 1	Event Group Conversion End Interrupt Enable. Please refer to <a href="#">Section 18.5.1</a> for more details on the conversion end interrupts. Any operation mode read/write: 0 No interrupt is generated when conversion of all the channels selected for conversion in the Event Group is done. 1 An Event Group conversion end interrupt is generated when conversion of all the channels selected for conversion in the Event Group is done.
2	Reserved	0	Reads return zeros, writes have no effect.
1	EV OVR INT EN	0 1	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 <a href="#">Section 18.5.3</a> . Any operation mode read/write: 0 No interrupt is generated if an Event Group memory overrun occurs. 1 An Event Group memory overrun interrupt is generated if an Event Group memory overrun condition occurs.
0	EV THR INT EN	0 1	Event Group Threshold Interrupt Enable. An Event Group threshold interrupt occurs when the programmed Event Group threshold counter counts down to zero. Please refer to <a href="#">Section 18.5.2</a> for more details. Any operation mode read/write: 0 No interrupt is generated if the Event Group threshold counter reaches zero. 1 An Event Group threshold interrupt is generated if the Event Group threshold counter reaches zero.

### 18.11.12 ADC Group1 Interrupt Enable Control Register (ADG1INTENA)

ADC Group1 Interrupt Enable Control Register (ADG1INTENA) is shown in [Figure 18-28](#) and described in [Table 18-18](#).

**Figure 18-28. ADC Group1 Interrupt Enable Control Register (ADG1INTENA)**

31	Reserved					16
R-0						
15	Reserved					8
R-0						
7	4	3	2	1	0	
Reserved		G1 END INT EN	Reserved	G1 OVR INT EN	G1 THR INT EN	
R-0		R/W-0	R-0	R/W-0	R/W-0	

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

**Table 18-18. ADC Group1 Interrupt Enable Control Register (ADG1INTENA) Field Descriptions**

Bit	Field	Value	Description
31-4	Reserved	0	Reads return zeros, writes have no effect.
3	G1 END INT EN	0	Group1 Conversion End Interrupt Enable. Please refer to <a href="#">Section 18.5.1</a> for more details on the conversion end interrupts. Any operation mode read/write: 0 No interrupt is generated when conversion of all the channels selected for conversion in the Group1 is done.
		1	A Group1 conversion end interrupt is generated when conversion of all the channels selected for conversion in the Group1 is done.
2	Reserved	0	Reads return zeros, writes have no effect.
1	G1 OVR INT EN	0	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 <a href="#">Section 18.5.3</a> . Any operation mode read/write: 0 No interrupt is generated if a Group1 memory overrun occurs.
		1	A Group1 memory overrun interrupt is generated if a Group1 memory overrun condition occurs.
0	G1 THR INT EN	0	Group1 Threshold Interrupt Enable. A Group1 threshold interrupt occurs when the programmed Group1 threshold counter counts down to zero. Please refer to <a href="#">Section 18.5.2</a> for more details. Any operation mode read/write: 0 No interrupt is generated if the Group1 threshold counter reaches zero.
		1	A Group1 threshold interrupt is generated if the Group1 threshold counter reaches zero.

### 18.11.13 ADC Group2 Interrupt Enable Control Register (ADG2INTENA)

ADC Group2 Interrupt Enable Control Register (ADG2INTENA) is shown in [Figure 18-29](#) and described in [Table 18-19](#).

**Figure 18-29. ADC Group2 Interrupt Enable Control Register (ADG2INTENA)**

31	Reserved				16
R-0					
15	Reserved				8
R-0					
7	4	3	2	1	0
Reserved	G2 END INT EN	Reserved	G2 OVR INT EN	G2 THR INT EN	
R-0	R/W-0	R-0	R/W-0	R/W-0	

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

**Table 18-19. ADC Group2 Interrupt Enable Control Register (ADG2INTENA) Field Descriptions**

Bit	Field	Value	Description
31-4	Reserved	0	Reads return zeros, writes have no effect.
3	G2 END INT EN	0 1	Group2 Conversion End Interrupt Enable. Please refer to <a href="#">Section 18.5.1</a> for more details on the conversion end interrupts. Any operation mode read/write: 0 No interrupt is generated when conversion of all the channels selected for conversion in the Group2 is done. 1 A Group2 conversion end interrupt is generated when conversion of all the channels selected for conversion in the Group2 is done.
2	Reserved	0	Reads return zeros, writes have no effect.
1	G2 OVR INT EN	0 1	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 <a href="#">Section 18.5.3</a> . Any operation mode read/write: 0 No interrupt is generated if a Group2 memory overrun occurs. 1 A Group2 memory overrun interrupt is generated if a Group2 memory overrun condition occurs.
0	G2 THR INT EN	0 1	Group2 Threshold Interrupt Enable. A Group2 threshold interrupt occurs when the programmed Group2 threshold counter counts down to zero. Please refer to <a href="#">Section 18.5.2</a> for more details. Any operation mode read/write: 0 No interrupt is generated if the Group2 threshold counter reaches zero. 1 A Group2 threshold interrupt is generated if the Group2 threshold counter reaches zero.

### 18.11.14 ADC Event Group Interrupt Flag Register (ADEVINTFLG)

ADC Event Group Interrupt Enable Control Register (ADEVINTENA) is shown in [Figure 18-30](#) and described in [Table 18-20](#).

**Figure 18-30. ADC Event Group Interrupt Flag Register (ADEVINTFLG)**

31	Reserved					16
R-0						
15	Reserved					8
R-0						
7	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	R-0	

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

**Table 18-20. ADC Event Group Interrupt Flag Register (ADEVINTFLG) Field Descriptions**

Bit	Field	Value	Description
31-4	Reserved	0	Reads return zeros, writes have no effect.
3	EV END	0 1	<p>Event Group Conversion End. This bit will be set only if the Event Group conversions are configured to be in the single-conversion mode.</p> <p>Any operation mode read:</p> <p>0 All the channels selected for conversion in the Event Group have not yet been converted.</p> <p>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.</p> <p>This bit can be cleared by any one of the following ways:</p> <ul style="list-style-type: none"> <li>• By writing a '1' to this bit</li> <li>• By writing a '1' to the Event Group status register bit 0 (EV END)</li> <li>• By reading one conversion result from the Event Group results' memory in the "read from FIFO" mode</li> <li>• By writing a new set of channels to the Event Group channel select register</li> </ul>
2	EV MEM EMPTY	0 1	<p>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.</p> <p>Any operation mode read:</p> <p>0 The Event Group results memory is not empty.</p> <p>1 The Event Group results memory is empty.</p>
1	EV MEM OVERRUN	0 1	<p>Event Group Memory Overrun. This is a read-only bit; writes have no effect.</p> <p>Any operation mode read:</p> <p>0 Event Group results memory has not overrun.</p> <p>1 Event Group results memory has overrun.</p>
0	EV THR INT FLG	0 1	<p>Event Group Threshold Interrupt Flag.</p> <p>Any operation mode read:</p> <p>0 The number of conversions completed for the Event Group is smaller than the threshold programmed in the Event Group interrupt threshold register.</p> <p>1 The number of conversions completed for the Event Group is equal to or greater than the threshold programmed in the Event Group interrupt threshold register.</p> <p>This bit can be cleared by writing a '1' ; writing a '0' has no effect.</p>

### 18.11.15 ADC Group1 Interrupt Flag Register (ADG1INTFLG)

ADC Group1 Interrupt Flag Register (ADG1INTFLG) is shown in [Figure 18-31](#) and described in [Table 18-21](#).

**Figure 18-31. ADC Group1 Interrupt Flag Register (ADG1INTFLG)**

31	Reserved				16
R-0					
15	Reserved				8
R-0					
7	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	R-0

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

**Table 18-21. ADC Group1 Interrupt Flag Register (ADG1INTFLG) Field Descriptions**

Bit	Field	Value	Description
31-4	Reserved	0	Reads return zeros, writes have no effect.
3	G1 END	0 1	<p>Group1 Conversion End. This bit will be set only if the Group1 conversions are configured to be in the single-conversion mode.</p> <p>Any operation mode read:</p> <p>0 All the channels selected for conversion in the Group1 have not yet been converted.</p> <p>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.</p> <p>This bit can be cleared by any one of the following ways:</p> <ul style="list-style-type: none"> <li>• By writing a '1' to this bit</li> <li>• By writing a '1' to the Group1 status register bit 0 (G1 END)</li> <li>• By reading one conversion result from the Group1 results' memory in the "read from FIFO" mode</li> <li>• By writing a new set of channels to the Group1 channel select register</li> </ul>
2	G1 MEM EMPTY	0 1	<p>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.</p> <p>Any operation mode read:</p> <p>0 The Group1 results memory is not empty.</p> <p>1 The Group1 results memory is empty.</p>
1	G1 MEM OVERRUN	0 1	<p>Group1 Memory Overrun. This is a read-only bit; writes have no effect.</p> <p>Any operation mode read:</p> <p>0 Group1 results memory has not overrun.</p> <p>1 Group1 results memory has overrun.</p>
0	G1 THR INT FLG	0 1	<p>Group1 Threshold Interrupt Flag.</p> <p>Any operation mode read:</p> <p>0 The number of conversions completed for the Group1 is smaller than the threshold programmed in the Group1 interrupt threshold register.</p> <p>1 The number of conversions completed for the Group1 is equal to or greater than the threshold programmed in the Group1 interrupt threshold register.</p> <p>This bit can be cleared by writing a '1' ; writing a '0' has no effect.</p>

### 18.11.16 ADC Group2 Interrupt Flag Register (ADG2INTFLG)

ADC Group2 Interrupt Flag Register (ADG2INTFLG) is shown in [Figure 18-32](#) and described in [Table 18-22](#).

**Figure 18-32. ADC Group2 Interrupt Flag Register (ADG2INTFLG)**

31	Reserved					16
R-0						
15	Reserved					8
R-0						
7	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	R-0	

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

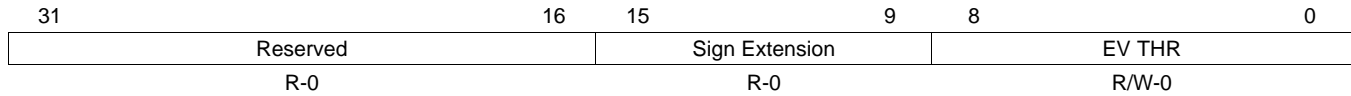
**Table 18-22. ADC Group2 Interrupt Flag Register (ADG2INTFLG) Field Descriptions**

Bit	Field	Value	Description
31-4	Reserved	0	Reads return zeros, writes have no effect.
3	G2 END	0 1	<p>Group2 Conversion End. This bit will be set only if the Group2 conversions are configured to be in the single-conversion mode.</p> <p>Any operation mode read:</p> <p>0 All the channels selected for conversion in the Group2 have not yet been converted.</p> <p>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.</p> <p>This bit can be cleared by any one of the following ways:</p> <ul style="list-style-type: none"> <li>• By writing a '1' to this bit</li> <li>• By writing a '1' to the Group2 status register bit 0 (G2 END)</li> <li>• By reading one conversion result from the Group2 results' memory in the "read from FIFO" mode</li> <li>• By writing a new set of channels to the Group2 channel select register</li> </ul>
2	G2 MEM EMPTY	0 1	<p>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.</p> <p>Any operation mode read:</p> <p>0 The Group2 results memory is not empty.</p> <p>1 The Group2 results memory is empty.</p>
1	G2 MEM OVERRUN	0 1	<p>Group2 Memory Overrun. This is a read-only bit; writes have no effect.</p> <p>Any operation mode read:</p> <p>0 Group2 results memory has not overrun.</p> <p>1 Group2 results memory has overrun.</p>
0	G2 THR INT FLG	0 1	<p>Group2 Threshold Interrupt Flag.</p> <p>Any operation mode read:</p> <p>0 The number of conversions completed for the Group2 is smaller than the threshold programmed in the Group2 interrupt threshold register.</p> <p>1 The number of conversions completed for the Group2 is equal to or greater than the threshold programmed in the Group2 interrupt threshold register.</p> <p>This bit can be cleared by writing a '1' ; writing a '0' has no effect.</p>

### 18.11.17 ADC Event Group Threshold Interrupt Control Register (ADEVTHRINTCR)

ADC Event Group Threshold Interrupt Control Register (ADEVTHRINTCR) is shown in [Figure 18-33](#) and described in [Table 18-23](#).

**Figure 18-33. ADC Event Group Threshold Interrupt Control Register (ADEVTHRINTCR)**



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

**Table 18-23. ADC Event Group Threshold Interrupt Control Register (ADEVTHRINTCR) Field Descriptions**

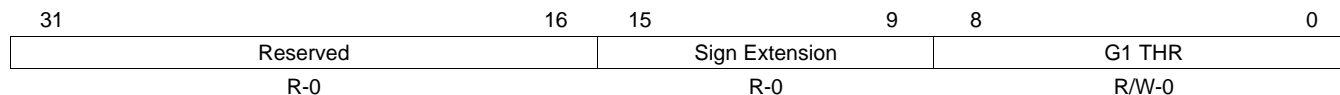
Bit	Field	Value	Description
31-16	Reserved	0	Reads return zeros, writes have no effect.
15-9	Sign Extension		These bits always read the same as the bit 8 of this register.
8-0	EV THR		<p>Event Group Threshold Counter.</p> <p>Before ADC conversions begin on the Event Group, this field is initialized to the number of conversion results that the Event Group memory should contain before interrupting the CPU. This counter decrements when the ADC module writes a new conversion result to the Event Group results' memory. The counter increments for each read of a conversion result from the Event Group results' memory in the "read from FIFO" mode. The threshold counter is not affected for a direct read from the Event Group results' memory. Also, a simultaneous ADC write and a CPU/DMA read from the Event Group FIFO will leave the threshold counter unchanged. In case of an Event Group Results' memory overrun condition, if new conversion results are not allowed to overwrite the existing memory contents, then the Event Group threshold counter is not decremented.</p> <p>Please refer to <a href="#">Section 18.5.2</a> for more details on the threshold interrupts.</p>



### 18.11.18 ADC Group1 Threshold Interrupt Control Register (ADG1THRINTCR)

ADC Group1 Threshold Interrupt Control Register (ADG1THRINTCR) is shown in [Figure 18-34](#) and described in [Table 18-24](#).

**Figure 18-34. ADC Group1 Threshold Interrupt Control Register (ADG1THRINTCR)**



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

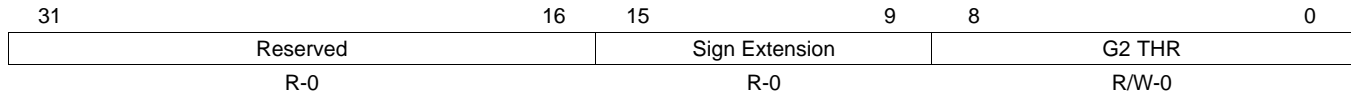
**Table 18-24. ADC Group1 Threshold Interrupt Control Register (ADG1THRINTCR) Field Descriptions**

Bit	Field	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		<p>Group1 Threshold Counter.</p> <p>Before ADC conversions begin on the Group1, this field is initialized to the number of conversion results that the Group1 memory should contain before interrupting the CPU. This counter decrements when the ADC module writes a new conversion result to the Group1 results' memory. The counter increments for each read of a conversion result from the Group1 results' memory in the "read from FIFO" mode. The threshold counter is not affected for a direct read from the group1 results' memory. Also, a simultaneous ADC write and a CPU/DMA read from the Group1 FIFO will leave the threshold counter unchanged. In case of an Group1 Results' memory overrun condition, if new conversion results are not allowed to overwrite the existing memory contents, then the Group1 threshold counter is not decremented.</p> <p>Please refer to <a href="#">Section 18.5.2</a> for more details on the threshold interrupts.</p>

### 18.11.19 ADC Group2 Threshold Interrupt Control Register (ADG2THRINTCR)

The ADC Group2 Threshold Interrupt Control Register (ADG2THRINTCR) is shown in [Figure 18-35](#) and described in [Table 18-25](#).

**Figure 18-35. ADC Group2 Threshold Interrupt Control Register (ADG2THRINTCR)**



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

**Table 18-25. ADC Group2 Threshold Interrupt Control Register (ADG2THRINTCR) Field Descriptions**

Bit	Field	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		<p>Group2 Threshold Counter.</p> <p>Before ADC conversions begin on the Group2, this field is initialized to the number of conversion results that the Group2 memory should contain before interrupting the CPU. This counter decrements when the ADC module writes a new conversion result to the Group2 results' memory. The counter increments for each read of a conversion result from the Group2 results' memory in the "read from FIFO" mode. The threshold counter is not affected for a direct read from the group2 results' memory. Also, a simultaneous ADC write and a CPU/DMA read from the Group2 FIFO will leave the threshold counter unchanged. In case of an Group2 Results' memory overrun condition, if new conversion results are not allowed to overwrite the existing memory contents, then the Group2 threshold counter is not decremented.</p> <p>Please refer to <a href="#">Section 18.5.2</a> for more details on the threshold interrupts.</p>

### 18.11.20 ADC Event Group DMA Control Register (ADEVDMACR)

ADC Event Group DMA Control Register (ADEVDMACR) is shown in [Figure 18-36](#) and described in [Table 18-26](#).

**Figure 18-36. ADC Event Group DMA Control Register (ADEVDMACR)**

31	25	24				16
Reserved			EV BLOCKS			
R-0			R/W-0			
15					8	
Reserved						
R-0						
7	4	3	2	1	0	
Reserved		DMA EV END	EV BLK XFER	Reserved	EV DMA EN	
R-0		R/W-0	R/W-0	R-0	R/W-0	

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

**Table 18-26. ADC Event Group DMA Control Register (ADEVDMACR) Field Descriptions**

Bit	Field	Value	Description
31-25	Reserved	0	Reads return zeros, writes have no effect.
24-16	EV BLOCKS	0 1h-1FFh	<p>Number of Event Group Result buffers to be transferred using DMA if the ADC module is configured to generate a DMA request. If the Event Group is configured to use the block transfer mode of the DMA module, then the ADC module generates a DMA request after the Event Group results' memory accumulates EV BLOCKS number of conversion results.</p> <p>This feature is designed to be used in place of the threshold interrupt for the Event Group. As a result, the EV THR field of the Event Group Interrupt Threshold Control Register and the EV BLOCKS field of the Event Group DMA Control Register are the same.</p> <p>Any operation mode read/write:</p> <p>0 No DMA transfer occurs even if EV BLK XFER is set to '1'.</p> <p>One DMA request is generated if the EV BLK XFER is set to '1' and the specified number of Event Group conversion results have been accumulated.</p>
15-4	Reserved	0	Reads return zeros, writes have no effect.
3	DMA EV END		<p>Event Group Conversion End DMA Transfer Enable.</p> <p>Any operation mode read:</p> <p>ADC module generates a DMA request for each write to the Event group results RAM if EV DMA EN is set.</p> <p>ADC module generates a DMA request when the ADC has completed the conversions for all channels selected for conversion in the event group.</p>
2	EV BLK XFER	0 1	<p>Event Group Block DMA Transfer Enable.</p> <p>Any operation mode read:</p> <p>0 ADC module generates a DMA request for each write to the Event Group memory if EV DMA EN is set.</p> <p>1 ADC module generates a DMA request when the ADC has written EV BLOCKS number of buffers into the Event Group memory.</p> <p>If EV BLK XFER bit is set to '1', EV DMA EN bit is ignored and DMA requests will be generated every time the Threshold Counter reaches '0' from a count value of 1.</p>
1	Reserved	0	Reads return zeros, writes have no effect.
0	EV DMA EN	0 1	<p>Event Group DMA Transfer Enable.</p> <p>Any operation mode read:</p> <p>0 ADC module does not generate a DMA request when it writes the conversion result to the Event Group memory.</p> <p>1 ADC module generates a DMA transfer when the ADC has written to the Event Group memory. The EV BLK XFER bit must be cleared to '0' for this DMA request to be generated.</p>

### 18.11.21 ADC Group1 DMA Control Register (ADG1DMACR)

ADC Group1 DMA Control Register (ADG1DMACR) is shown in [Figure 18-37](#) and described in [Table 18-27](#).

**Figure 18-37. ADC Group1 DMA Control Register (ADG1DMACR)**



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

**Table 18-27. ADC Group1 DMA Control Register (ADG1DMACR) Field Descriptions**

Bit	Field	Value	Description
31-25	Reserved	0	Reads return zeros, writes have no effect.
24-16	G1 BLOCKS	0 1h-1FFh	<p>Number of Group1 Result buffers to be transferred using DMA if the ADC module is configured to generate a DMA request. If the Group1 is configured to use the block transfer mode of the DMA module, then the ADC module generates a DMA request after the Group1 results' memory accumulates G1 BLOCKS number of conversion results.</p> <p>This feature is designed to be used in place of the threshold interrupt for the Group1. As a result, the G1 THR field of the Group1 Interrupt Threshold Control Register and the G1 BLOCKS field of the Group1 DMA Control Register are the same.</p> <p>Any operation mode read/write:</p> <p>0 No DMA transfer occurs even if G1 BLK XFER is set to '1'.</p> <p>One DMA request is generated if the G1 BLK XFER is set to '1' and the specified number of Group1 conversion results have been accumulated.</p>
15-4	Reserved	0	Reads return zeros, writes have no effect.
3	DMA G1 END		<p>Group1 Conversion End DMA Transfer Enable.</p> <p>Any operation mode read:</p> <p>ADC module generates a DMA request for each write to the group1 results RAM if G1 DMA EN is set.</p> <p>ADC module generates a DMA request when the ADC has completed the conversions for all channels selected for conversion in the group1.</p>
2	G1 BLK XFER	0 1	<p>Group1 Block DMA Transfer Enable.</p> <p>Any operation mode read:</p> <p>0 ADC module generates a DMA request for each write to the Group1 memory if G1 DMA EN is set.</p> <p>1 ADC module generates a DMA request when the ADC has written G1 BLOCKS number of buffers into the Group1 memory.</p> <p>If G1 BLK XFER bit is set to '1', G1 DMA EN bit is ignored and DMA requests will be generated every time the Threshold Counter reaches '0' from a count value of 1.</p>
1	Reserved	0	Reads return zeros, writes have no effect.
0	G1 DMA EN	0 1	<p>Group1 DMA Transfer Enable.</p> <p>Any operation mode read:</p> <p>0 ADC module does not generate a DMA request when it writes the conversion result to the Group1 memory.</p> <p>1 ADC module generates a DMA transfer when the ADC has written to the Group1 memory. The G1 BLK XFER bit must be cleared to '0' for this DMA request to be generated.</p>

### 18.11.22 ADC Group2 DMA Control Register (ADG2DMACR)

ADC Group2 DMA Control Register (ADG2DMACR) is shown in [Figure 18-38](#) and described in [Table 18-28](#).

**Figure 18-38. ADC Group2 DMA Control Register (ADG2DMACR)**

31	25	24				16
Reserved			G2 BLOCKS			
R-0			R/W-0			
15						8
Reserved						
R-0						
7	4	3	2	1	0	
Reserved		DMA G2 END	G2 BLK XFER	Reserved	G2 DMA EN	
R-0		R/W-0	R/W-0	R-0	R/W-0	

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

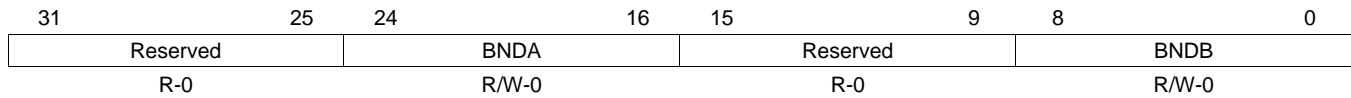
**Table 18-28. ADC Group2 DMA Control Register (ADG2DMACR) Field Descriptions**

Bit	Field	Value	Description
31-25	Reserved	0	Reads return zeros, writes have no effect.
24-16	G2 BLOCKS	0 1h-1FFh	<p>Number of Group2 Result buffers to be transferred using DMA if the ADC module is configured to generate a DMA request. If the Group2 is configured to use the block transfer mode of the DMA module, then the ADC module generates a DMA request after the Group2 results' memory accumulates G2 BLOCKS number of conversion results.</p> <p>This feature is designed to be used in place of the threshold interrupt for the Group2. As a result, the G2 THR field of the Group2 Interrupt Threshold Control Register and the G2 BLOCKS field of the Group2 DMA Control Register are the same.</p> <p>Any operation mode read/write:</p> <p>0 No DMA transfer occurs even if G2 BLK XFER is set to '1'.</p> <p>One DMA request is generated if the G2 BLK XFER is set to '1' and the specified number of Group2 conversion results have been accumulated.</p>
15-4	Reserved	0	Reads return zeros, writes have no effect.
3	DMA G2 END		<p>Group2 Conversion End DMA Transfer Enable.</p> <p>Any operation mode read:</p> <p>ADC module generates a DMA request for each write to the group2 results RAM if G2 DMA EN is set.</p> <p>ADC module generates a DMA request when the ADC has completed the conversions for all channels selected for conversion in the group2.</p>
2	G2 BLK XFER	0 1	<p>Group2 Block DMA Transfer Enable.</p> <p>Any operation mode read:</p> <p>0 ADC module generates a DMA request for each write to the Group2 memory if G2 DMA EN is set.</p> <p>1 ADC module generates a DMA request when the ADC has written G2 BLOCKS number of buffers into the Group2 memory.</p> <p>If G2 BLK XFER bit is set to '1', G2 DMA EN bit is ignored and DMA requests will be generated every time the Threshold Counter reaches '0' from a count value of 1.</p>
1	Reserved	0	Reads return zeros, writes have no effect.
0	G2 DMA EN	0 1	<p>Group2 DMA Transfer Enable.</p> <p>Any operation mode read:</p> <p>0 ADC module does not generate a DMA request when it writes the conversion result to the Group2 memory.</p> <p>1 ADC module generates a DMA transfer when the ADC has written to the Group2 memory. The G2 BLK XFER bit must be cleared to '0' for this DMA request to be generated.</p>

### 18.11.23 ADC Results Memory Configuration Register (ADBNDCCR)

ADC Results Memory Configuration Register (ADBNDCCR) [offset = 0x58] is shown in [Figure 18-39](#) and described in [Table 18-29](#).

**Figure 18-39. ADC Results Memory Configuration Register (ADBNDCCR)**



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

**Table 18-29. ADC Results Memory Configuration Register (ADBNDCCR) Field Descriptions**

Bit	Field	Value	Description
31-25	Reserved	0	Reads return zeros, writes have no effect.
24-16	BND A	0  0-1FFh	<p>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.</p> <p>Any operation mode read/write:</p> <p>0 Event Group conversions are not required. If Event Group conversions are performed with the BND A value of zero, then the Event Group memory size will default to 1024 words. For proper usage of the ADC results memory, configure the BND A value to be non-zero and lower than the BND B value.</p> <p>0-1FFh A total of (2 * BND A) buffers are available in the ADC results memory for storing Event Group conversion results.</p>
15-9	Reserved	0	Reads return zeros, writes have no effect.
8-0	BND B	0  0-1FFh	<p>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 BND A from BND B. As a result, BND B must always be specified as greater than or equal to BND A.</p> <p>Any operation mode read/write:</p> <p>0 Event Group as well as Group1 conversions are not required.</p> <p>0-1FFh A total of 2 * (BND B - BND A) buffers are available in the ADC results memory for storing Group1 conversion results.</p>

Please refer to [Section 18.3.8](#) for further details on how the conversion results are stored in the ADC results' RAM.

### 18.11.24 ADC Results Memory Size Configuration Register (ADBNDEND)

ADC Results Memory Size Configuration Register (ADBNDEND) is shown in [Figure 18-40](#) and described in [Table 18-30](#).

**Figure 18-40. ADC Results Memory Size Configuration Register (ADBNDEND)**

31	17	16	15	3	2	0
Reserved		BUF INIT ACTIVE	Reserved		BNDEND	
R-0		R-0	R-0		R/W-0	

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

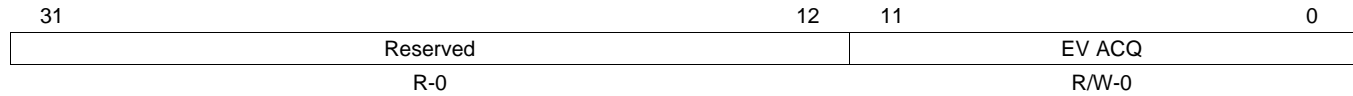
**Table 18-30. ADC Results Memory Size Configuration Register (ADBNDEND) Field Descriptions**

Bit	Field	Value	Description
31-17	Reserved	0	Reads return zeros, writes have no effect.
16	BUF INIT ACTIVE	0	ADC Results Memory Auto-initialization Status. Any operation mode read/write: 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.
2-0	BNDEND	0	Buffer Boundary End. These bits specify the total number of memory buffers available for storing the ADC conversion results. These bits should be programmed to match the number of ADC conversion result buffers required to be used for the application. Any operation mode read/write: 16 words available for storing ADC conversion results.
		1h	32 words available for storing ADC conversion results.
		2h	64 words available for storing ADC conversion results. This is the maximum configuration allowed since the device supports 64 buffers each for ADC1 as well as ADC2.
		4h-7h	Reserved. These combinations must not be used.

### 18.11.25 ADC Event Group Sampling Time Configuration Register (ADEVSAAMP)

ADC Event Group Sampling Time Configuration Register (ADEVSAAMP) is shown in [Figure 18-41](#) and described in [Table 18-31](#).

**Figure 18-41. ADC Event Group Sampling Time Configuration Register (ADEVSAAMP)**



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

**Table 18-31. ADC Event Group Sampling Time Configuration Register (ADEVSAAMP) Field Descriptions**

Bit	Field	Value	Description
31-12	Reserved	0	Reads return zeros, writes have no effect.
11-0	EV ACQ		<p>Event Group Acquisition Time. These bits define the sampling window (SW) for the Event Group conversions.</p> <p><math>SW = EV\ ACQ + 2</math> in terms of ADCLK cycles.</p> <p>There are two factors that determine the minimum sampling window value required:</p> <p>First, the ADC module design requires that <math>SW \geq 3</math> ADCLK cycles.</p> <p>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.</p>

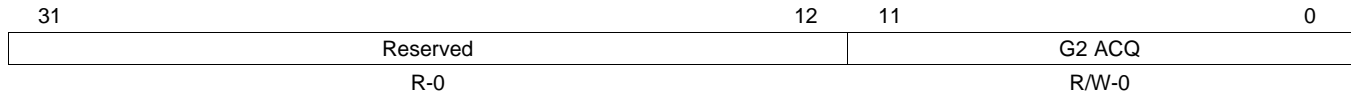




### 18.11.27 ADC Group2 Sampling Time Configuration Register (ADG2SAMP)

ADC Group2 Sampling Time Configuration Register (ADG2SAMP) is shown in [Figure 18-43](#) and described in [Table 18-33](#).

**Figure 18-43. ADC Group2 Sampling Time Configuration Register (ADG2SAMP)**



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

**Table 18-33. ADC Group2 Sampling Time Configuration Register (ADG2SAMP) Field Descriptions**

Bit	Field	Value	Description
31-12	Reserved	0	Reads return zeros, writes have no effect.
11-0	G2 ACQ		<p>Group2 Acquisition Time. These bits define the sampling window (SW) for the Group2 conversions. <math>SW = G2\ ACQ + 2</math> in terms of ADCLK cycles.</p> <p>There are two factors that determine the minimum sampling window value required:</p> <p>First, the ADC module design requires that <math>SW \geq 3</math> ADCLK cycles.</p> <p>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.</p>

### 18.11.28 ADC Event Group Status Register (ADEVSR)

ADC Event Group Status Register (ADEVSR) is shown in [Figure 18-44](#) and described in [Table 18-34](#).

**Figure 18-44. ADC Event Group Status Register (ADEVSR)**

31	4	3	2	1	0
Reserved		EV MEM EMPTY	EV BUSY	EV STOP	EV END
R-0		R-1	R-0	R-0	R/W-0

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

**Table 18-34. ADC Event Group Status Register (ADEVSR) Field Descriptions**

Bit	Field	Value	Description
31-4	Reserved	0	Reads return zeros, writes have no effect.
3	EV MEM EMPTY	0 The Event Group results memory has valid conversion results. 1 The Event Group results memory is empty, or does not contain any unread conversion results.	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.  Any operation mode read:
2	EV BUSY	0 Event Group conversions are neither in progress nor frozen. 1 Event Group conversions are either in progress, or are frozen for servicing some other group. This bit will always be set when the Event Group is configured to be in the continuous conversion mode.	Event Group Conversion Busy.  Any operation mode read:
1	EV STOP	0 Event Group conversions are not currently frozen. 1 Event Group conversions are currently frozen.	Event Group Conversion Stopped.  Any operation mode read:
0	EV END	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: <ul style="list-style-type: none"> <li>• By reading a conversion result from the Event Group results memory in the "read from FIFO" mode.</li> <li>• By writing a new value to the Event Group channel select register ADEVSEL.</li> <li>• By writing a '1' to this bit.</li> <li>• By disabling the ADC module by clearing the ADC EN bit.</li> </ul>	Event Group Conversions Ended.  Any operation mode read:

### 18.11.29 ADC Group1 Status Register (ADG1SR)

ADC Group1 Status Register (ADG1SR) is shown in [Figure 18-45](#) and described in [Table 18-35](#).

**Figure 18-45. ADC Group1 Status Register (ADG1SR)**

31	4	3	2	1	0
Reserved		G1 MEM EMPTY	G1 BUSY	G1 STOP	G1 END
R-0		R-1	R-0	R-0	R/W-0

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

**Table 18-35. ADC Group1 Status Register (ADG1SR) Field Descriptions**

Bit	Field	Value	Description
31-4	Reserved	0	Reads return zeros, writes have no effect.
3	G1 MEM EMPTY	0 1	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.  Any operation mode read: 0 The Group1 results memory has valid conversion results. 1 The Group1 results memory is empty, or does not contain any unread conversion results.
2	G1 BUSY	0 1	Group1 Conversion Busy.  Any operation mode read: 0 Group1 conversions are neither in progress nor frozen. 1 Group1 conversions are either in progress, or are frozen 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	0 1	Group1 Conversion Stopped.  Any operation mode read: 0 Group1 conversions are not currently frozen. 1 Group1 conversions are currently frozen.
0	G1 END	0 1	Group1 Conversions Ended.  Any operation 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: <ul style="list-style-type: none"> <li>• By reading a conversion result from the Group1 results memory in the "read from FIFO" mode.</li> <li>• By writing a new value to the Group1 channel select register ADG1SEL.</li> <li>• By writing a '1' to this bit.</li> <li>• By disabling the ADC module by clearing the ADC EN bit.</li> </ul>

### 18.11.30 ADC Group2 Status Register (ADG2SR)

ADC Group2 Status Register (ADG2SR) is shown in [Figure 18-46](#) and described in [Table 18-36](#).

**Figure 18-46. ADC Group2 Status Register (ADG2SR)**

31	4	3	2	1	0
Reserved		G2 MEM EMPTY	G2 BUSY	G2 STOP	G2 END
R-0		R-1	R-0	R-0	R/W-0

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

**Table 18-36. ADC Group2 Status Register (ADG2SR) Field Descriptions**

Bit	Field	Value	Description
31-4	Reserved	0	Reads return zeros, writes have no effect.
3	G2 MEM EMPTY	0 1	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.  Any operation mode read: 0 The Group2 results memory has valid conversion results. 1 The Group2 results memory is empty, or does not contain any unread conversion results.
2	G2 BUSY	0 1	Group2 Conversion Busy.  Any operation mode read: 0 Group2 conversions are neither in progress nor frozen. 1 Group2 conversions are either in progress, or are frozen 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	0 1	Group2 Conversion Stopped.  Any operation mode read: 0 Group2 conversions are not currently frozen. 1 Group2 conversions are currently frozen.
0	G2 END	0 1	Group2 Conversions Ended.  Any operation 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: <ul style="list-style-type: none"> <li>• By reading a conversion result from the Group2 results memory in the "read from FIFO" mode.</li> <li>• By writing a new value to the Group2 channel select register ADG2SEL.</li> <li>• By writing a '1' to this bit.</li> <li>• By disabling the ADC module by clearing the ADC EN bit.</li> </ul>

### 18.11.31 ADC Event Group Channel Select Register (ADEVSEL)

ADC Event Group Channel Select Register (ADEVSEL) is shown in [Figure 18-47](#) and described in [Table 18-37](#).

**Figure 18-47. ADC Event Group Channel Select Register (ADEVSEL)**

31	24	23	0
Reserved		V SEL	
R-0		R/W-0	

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

**Table 18-37. ADC Event Group Channel Select Register (ADEVSEL) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reads return zeros, writes have no effect.
15-0	EV SEL	0	Event Group channels selected. Any operation mode read/write: No ADC input channel is selected for conversion in the Event Group.
		Non-zero	The channels marked by the bit positions that are set to '1' 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 or DMA Threshold Counter again so that correct number of conversions happen before a Threshold interrupt or Block DMA request is generated.

---

ADC1 supports up to 24 channels and ADC2 supports up to 16 channels on the microcontroller.

### 18.11.32 ADC Group1 Channel Select Register (ADG1SEL)

ADC Group1 Channel Select Register (ADG1SEL) is shown in [Figure 18-48](#) and described in [Table 18-38](#).

**Figure 18-48. ADC Group1 Channel Select Register (ADG1SEL)**

31	24	23	0
Reserved		G1 SEL	
R-0		R/W-0	

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

**Table 18-38. ADC Group1 Channel Select Register (ADG1SEL) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reads return zeros, writes have no effect.
15-0	G1 SEL	0	Group1 channels selected. Any operation mode read/write: No ADC input channel is selected for conversion in the Group1.
		Non-zero	The channels marked by the bit positions that are set to '1' 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 or DMA Threshold Counter again so that correct number of conversions happen before a Threshold interrupt or Block DMA request is generated.

---

ADC1 supports up to 24 channels and ADC2 supports up to 16 channels on the microcontroller.

### 18.11.33 ADC Group2 Channel Select Register (ADG2SEL)

ADC Group2 Channel Select Register (ADG2SEL) is shown in [Figure 18-49](#) and described in [Table 18-39](#).

**Figure 18-49. ADC Group2 Channel Select Register (ADG2SEL)**

31	24	23	0
Reserved		G2 SEL	
R-0		R/W-0	

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

**Table 18-39. ADC Group2 Channel Select Register (ADG2SEL) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reads return zeros, writes have no effect.
15-0	G2 SEL	0	Group2 channels selected. Any operation mode read/write: No ADC input channel is selected for conversion in the Group2.
		Non-zero	The channels marked by the bit positions that are set to '1' 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 DMA Threshold Counter again so that correct number of conversions happen before a Threshold interrupt or Block DMA request is generated.

---

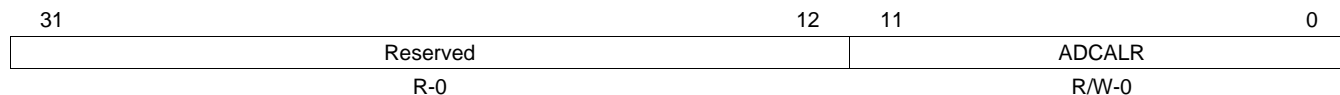
ADC1 supports up to 24 channels and ADC2 supports up to 16 channels on the microcontroller.



### 18.11.34 ADC Calibration and Error Offset Correction Register (ADCALR)

ADC Calibration and Error Offset Correction Register (ADCALR) is shown in [Figure 18-50](#) and described in [Table 18-40](#).

**Figure 18-50. ADC Calibration and Error Offset Correction Register (ADCALR)**



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

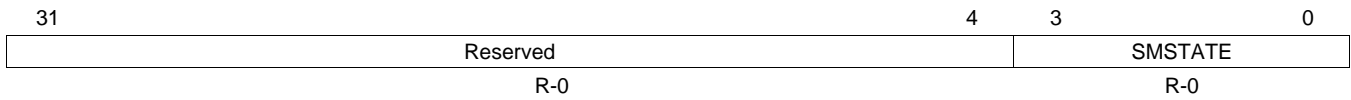
**Table 18-40. ADC Calibration and Error Offset Correction Register (ADCALR) Field Descriptions**

Bit	Field	Value	Description
31-12	Reserved	0	Reads return zeros, writes have no effect.
11-0	ADCALR	0-FFFh	<p>ADC Calibration Result and Offset Error Correction Value.</p> <p>The actual size of the ADCALR field is 12 or 10 bits depending on whether the ADC is configured to have a 12-, or 10-bit resolution respectively.</p> <p>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 2's complement form.</p> <p>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 <a href="#">Section 18.8.1</a>.</p>

### 18.11.35 ADC State Machine Status Register (ADSMSTATE)

Figure 18-51 and Table 18-41 describe the ADSMSTATE register.

**Figure 18-51. ADC State Machine Status Register (ADSMSTATE)**



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

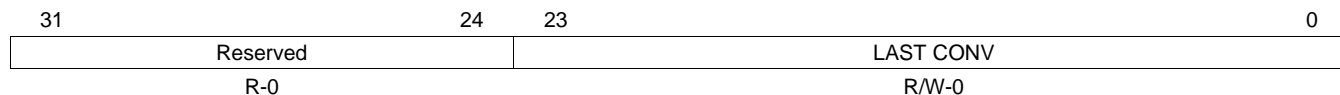
**Table 18-41. ADC State Machine Status Register (ADSMSTATE) Field Descriptions**

Bit	Field	Value	Description
31-4	Reserved	0	Reads return zeros, writes have no effect.
3-0	SM STATE		ADC State Machine Current State. These bits reflect the current state of the state machine and are reserved for use by TI for debug purposes.

### 18.11.36 ADC Channel Last Conversion Value Register (ADLASTCONV)

ADC Channel Last Conversion Value Register (ADLASTCONV) is shown in [Figure 18-52](#) and described in [Table 18-42](#).

**Figure 18-52. ADC Channel Last Conversion Value Register (ADLASTCONV)**



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

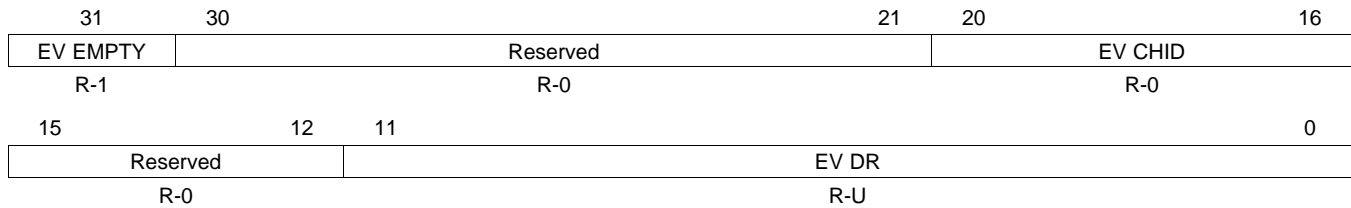
**Table 18-42. ADC Channel Last Conversion Value Register (ADLASTCONV) Field Descriptions**

Bit	Field	Value	Description
31-24	Reserved	0	Reads return zeros, writes have no effect.
23-0	LAST CONV	0	<p>ADC Input Channel's Last Converted Value.</p> <p>This register indicates whether the last converted value for a particular input channel was lower or higher than the mid-point of the reference voltage. In other words, this register acts as a digital input register and can be read by the application to determine the digital level at the input pins.</p> <p>This data is only valid for an input channel if it has been converted at least once.</p> <p>Any operation mode read for each bit of this register:</p>
		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.

**18.11.37 ADC Event Group Results' FIFO Register (ADEVBUFFER)**

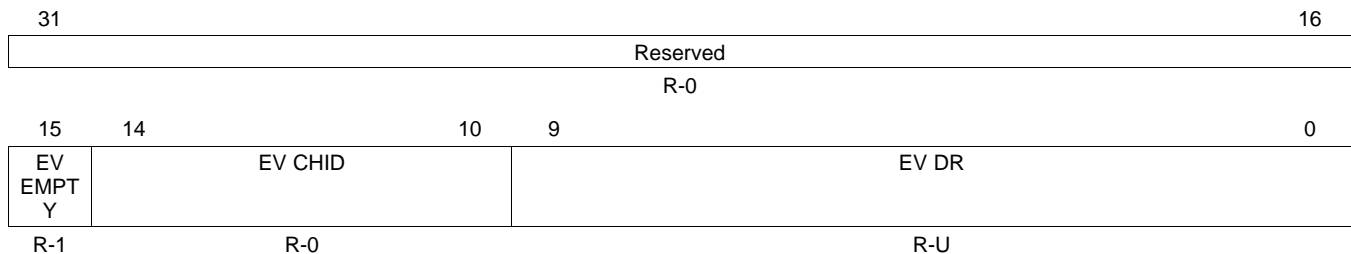
ADC Event Group Results' FIFO Register (ADEVBUFFER) is shown in [Figure 18-53](#) and described in [Table 18-43](#). As shown, the format of the data read from the AD EV BUFFER locations is different based on whether the ADC module is configured to be a 12-bit or a 10-bit ADC module.

**Figure 18-53. 12-bit ADC Event Group Results' FIFO Register (ADEVBUFFER)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset; -U = value after reset is unknown

**Figure 18-54. 10-bit ADC Event Group Results' FIFO Register (ADEVBUFFER)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset; -U = value after reset is unknown

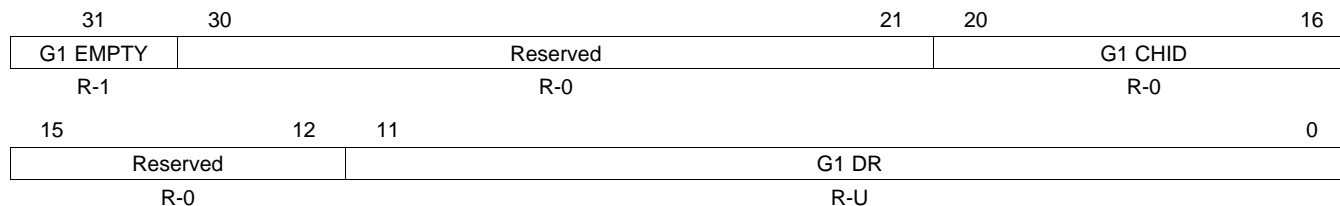
**Table 18-43. ADC Event Group Results' FIFO Register (ADEVBUFFER) Field Descriptions**

Bit	Field	Value	Description
	Reserved	0	Reads return zeros, writes have no effect.
	EV EMPTY	0 1	Event Group FIFO Empty. This bit is applicable only when the "read from FIFO" mode is used for reading the Event Group conversion results.  Any operation mode read: 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.
	EV CHID	0 1h-17h	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.  Any operation 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). 1h-17h The conversion result in the EV DR field of this buffer is from the ADC input channel number denoted by the EV CHID field.
	EV DR		Event Group Digital Conversion Result.  The Event Group results' FIFO location is aliased eight times, so that any word-aligned read from the address range 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.

### 18.11.38 ADC Group1 Results FIFO Register (ADG1BUFFER)

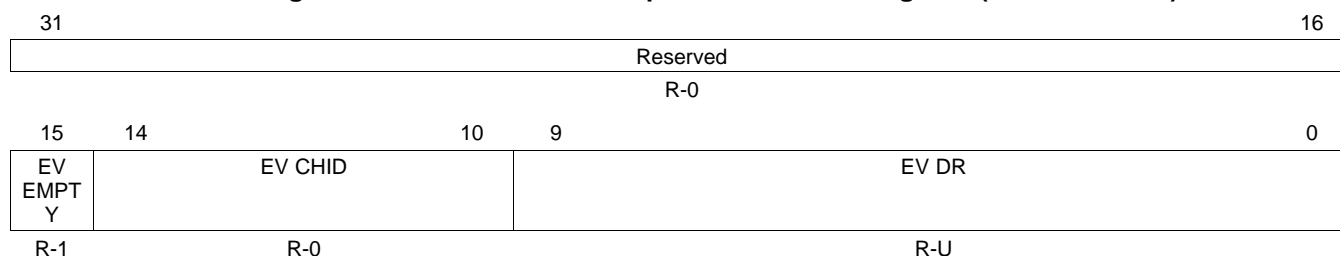
ADC Group1 Results FIFO Register (ADG1BUFFER) is shown in [Figure 18-55](#) and described in [Table 18-44](#). As shown, the format of the data read from the AD G1 BUFFER locations is different based on whether the ADC module is configured to be a 12-bit or a 10-bit ADC module.

**Figure 18-55. 12-bit ADC Group1 Results FIFO Register (ADG1BUFFER)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset; -U = value after reset is unknown

**Figure 18-56. 10-bit ADC Group1 Results' FIFO Register (ADG1BUFFER)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset; -U = value after reset is unknown

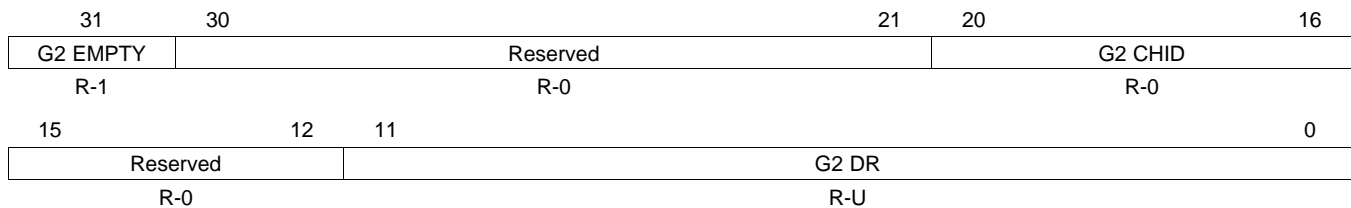
**Table 18-44. ADC Group1 Results FIFO Register (ADG1BUFFER) Field Descriptions**

Bit	Field	Value	Description
	Reserved	0	Reads return zeros, writes have no effect.
	G1 EMPTY	0 1	Group1 FIFO Empty. This bit is applicable only when the "read from FIFO" mode is used for reading the Group1 conversion results.  Any operation mode read: 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.
	G1 CHID	0 1h-17h	Group1 Channel Id. These bits are also applicable only when the "read from FIFO" mode is used for reading the Group1 conversion results.  Any operation 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). 1h-17h The conversion result in the G1 DR field of this buffer is from the ADC input channel number denoted by the G1 CHID field.
	G1 DR		Group1 Digital Conversion Result.  The Group1 results' FIFO location is aliased eight times, so that any 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.

### 18.11.39 ADC Group2 Results FIFO Register (ADG2BUFFER)

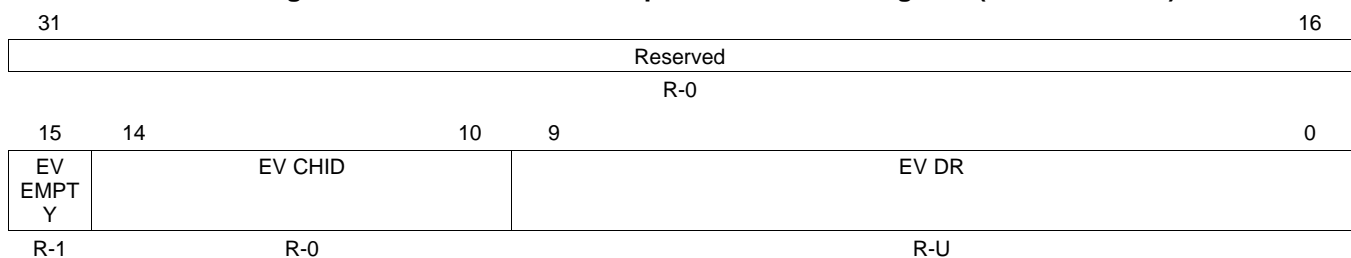
ADC Group2 Results FIFO Register (ADG2BUFFER) is shown in [Figure 18-57](#) and described in [Table 18-45](#). As shown, the format of the data read from the AD G2 BUFFER locations is different based on whether the ADC module is configured to be a 12-bit or a 10-bit ADC module.

**Figure 18-57. 12-bit ADC Group2 Results FIFO Register (ADG2BUFFER)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset; -U = value after reset is unknown

**Figure 18-58. 10-bit ADC Group2 Results' FIFO Register (ADG2BUFFER)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset; -U = value after reset is unknown

**Table 18-45. ADC Group2 Results FIFO Register (ADG2BUFFER) Field Descriptions**

Bit	Field	Value	Description
	Reserved	0	Reads return zeros, writes have no effect.
	G2 EMPTY	0 1	Group2 FIFO Empty. This bit is applicable only when the "read from FIFO" mode is used for reading the Group2 conversion results.  Any operation mode read: The data in the G2 DR field of this buffer is valid. The data in the G2 DR field of this buffer is not valid and there are no valid data in the Group2 results memory.
	G2 CHID	0 1h-17h	Group2 Channel Id. These bits are also applicable only when the "read from FIFO" mode is used for reading the Group2 conversion results.  Any operation mode read: 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). The conversion result in the G2 DR field of this buffer is from the ADC input channel number denoted by the G2 CHID field.
	G2 DR		Group2 Digital Conversion Result.  The Group2 results' FIFO location is aliased eight times, so that any 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.

**18.11.40 ADC Event Group Results Emulation FIFO Register (ADEVEMUBUFFER)**

A read from this location also gives out one conversion result from the Event Group results' memory along with the "EV EMPTY" status bit and the optional channel id. However, this read will not affect any of the status flags in the Event Group interrupt flag register or the Event Group status register. This register is useful for debuggers.

**18.11.41 ADC Group1 Results Emulation FIFO Register (ADG1EMUBUFFER)**

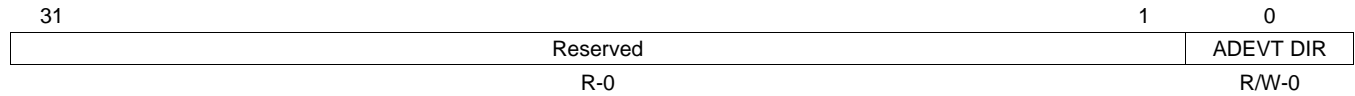
A read from this location also gives out one conversion result from the Group1 results' memory along with the "G1 EMPTY" status bit and the optional channel id. However, this read will not affect any of the status flags in the Group1 interrupt flag register or the Group1 status register. This register is useful for debuggers.

**18.11.42 ADC Group2 Results Emulation FIFO Register (ADG2EMUBUFFER)**

A read from this location also gives out one conversion result from the Group2 results' memory along with the "G2 EMPTY" status bit and the optional channel id. However, this read will not affect any of the status flags in the Group2 interrupt flag register or the Group2 status register. This register is useful for debuggers.

**18.11.43 ADC ADEVT Pin Direction Control Register (ADEVTDIR)**

ADC ADEVT Pin Direction Control Register (ADEVTDIR) is shown in [Figure 18-59](#) and described in [Table 18-46](#).

**Figure 18-59. ADC ADEVT Pin Direction Control Register (ADEVTDIR)**


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

**Table 18-46. ADC ADEVT Pin Direction Control Register (ADEVTDIR) Field Descriptions**

Bit	Field	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	ADEVT is an input pin; the output buffer is disabled.
		1	ADEVT is an output pin; the output buffer is enabled.



### 18.11.44 ADC ADEVT Pin Output Value Control Register (ADEVTOUT)

ADC ADEVT Pin Output Value Control Register (ADEVTOUT) is shown in [Figure 18-60](#) and described in [Table 18-47](#).

**Figure 18-60. ADC ADEVT Pin Output Value Control Register (ADEVTOUT)**

31	Reserved	1	0
R-0			ADEVT OUT R/W-0

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

**Table 18-47. ADC ADEVT Pin Output Value Control Register (ADEVTOUT) Field Descriptions**

Bit	Field	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	Output logic LOW on the ADEVT pin.
		1	Output logic HIGH on the ADEVT pin.

### 18.11.45 ADC ADEVT Pin Input Value Register (ADEVTIN)

ADC ADEVT Pin Input Value Register (ADEVTIN) is shown in [Figure 18-61](#) and described in [Table 18-48](#).

**Figure 18-61. ADC ADEVT Pin Input Value Register (ADEVTIN)**

31	Reserved	1	0
R-0		R-U	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset; -U = value after reset is unknown

**Table 18-48. ADC ADEVT Pin Input Value Register (ADEVTIN) Field Descriptions**

Bit	Field	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	Logic LOW present on the ADEVT pin.
		1	Logic HIGH present on the ADEVT pin.

### 18.11.46 ADC ADEVT Pin Set Register (ADEVTSET)

ADC ADEVT Pin Set Register (ADEVTSET) is shown in [Figure 18-62](#) and described in [Table 18-49](#).

**Figure 18-62. ADC ADEVT Pin Set Register (ADEVTSET)**

31	Reserved	1	0
R-0			ADEVT SET R/W-0

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

**Table 18-49. ADC ADEVT Pin Set Register (ADEVTSET) Field Descriptions**

Bit	Field	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. A read from this bit always returns the current state of the ADEVT pin.
			Any operating mode read/write:
		0	Output value on the ADEVT pin is unchanged.
		1	Output logic HIGH on the ADEVT pin, if the pin is configured to be an output pin.

**18.11.47 ADC ADEVT Pin Clear Register (ADEVTCLR)**

ADC ADEVT Pin Clear Register (ADEVTCLR) is shown in [Figure 18-63](#) and described in [Table 18-50](#).

**Figure 18-63. ADC ADEVT Pin Clear Register (ADEVTCLR)**

31	Reserved	1	0
R-0			ADEVT CLR R/W-0

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

**Table 18-50. ADC ADEVT Pin Clear Register (ADEVTCLR) Field Descriptions**

Bit	Field	Value	Description
31-1	Reserved	0	Reads return zeros, writes have no effect.
0	ADEVT CLR		ADEVT Pin Clear. A read from this bit always returns the current state of the ADEVT pin. Any operating mode read/write:
		0	Output value on the ADEVT pin is unchanged.
		1	Output logic LOW on the ADEVT pin, if the pin is configured to be an output pin.

### 18.11.48 ADC ADEVT Pin Open Drain Enable Register (ADEVTPDR)

ADC ADEVT Pin Open Drain Enable Register (ADEVTPDR) is shown in [Figure 18-64](#) and described in [Table 18-51](#).

**Figure 18-64. ADC ADEVT Pin Open Drain Enable Register (ADEVTPDR)**

31	Reserved	1	0
R-0			ADEVT PDR R/W-0

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

**Table 18-51. ADC ADEVT Pin Open Drain Enable Register (ADEVTPDR) Field Descriptions**

Bit	Field	Value	Description
31-1	Reserved	0	Reads return zeros, writes have no effect.
0	ADEVT PDR		ADEVT Pin Open Drain Enable. This bit enables the open-drain capability for the ADEVT pin if it is configured to be an output and a logic HIGH is being driven on to the pin. Any operating mode read/write:
		0	Output value on the ADEVT pin is logic HIGH.
		1	The ADEVT pin is tristated.

**18.11.49 ADC ADEVT Pin Pull Control Disable Register (ADEVTPDIS)**

ADC ADEVT Pin Pull Control Disable Register (ADEVTPDIS) is shown in [Figure 18-65](#) and described in [Table 18-52](#).

**Figure 18-65. ADC ADEVT Pin Pull Control Disable Register (ADEVTPDIS)**

31	Reserved	1	0
R-0			ADEVTPDIS R/W-0

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

**Table 18-52. ADC ADEVT Pin Pull Control Disable Register (ADEVTPDIS) Field Descriptions**

Bit	Field	Value	Description
31-1	Reserved	0	Reads return zeros, writes have no effect.
0	ADEVTPDIS		ADEVTPDIS Pin Pull Control Disable. This bit enables or disables the pull control on the ADEVTPDIS pin if it is configured to be an input pin.
		0	Any operating mode read/write: Pull on ADEVTPDIS pin is enabled.
		1	Pull on ADEVTPDIS pin is disabled.

### 18.11.50 ADC ADEVT Pin Pull Control Select Register (ADEVTPSEL)

ADC ADEVT Pin Pull Control Select Register (ADEVTPSEL) is shown in [Figure 18-66](#) and described in [Table 18-53](#).

**Figure 18-66. ADC ADEVT Pin Pull Control Select Register (ADEVTPSEL)**

31	Reserved	1	0
	R-0		ADEVT PSEL R/W-0

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

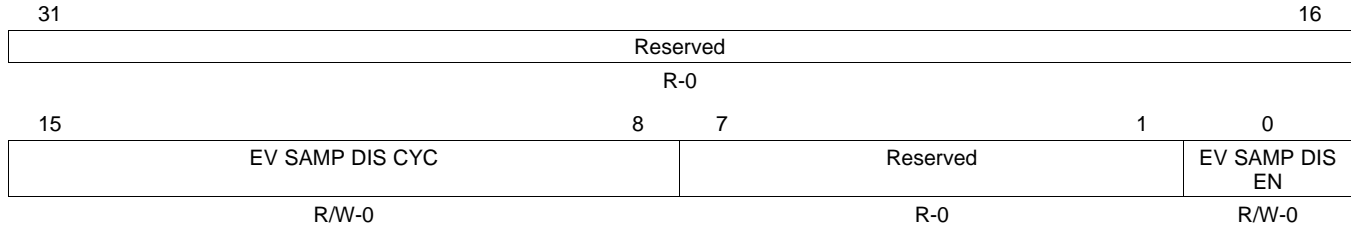
**Table 18-53. ADC ADEVT Pin Pull Control Select Register (ADEVTPSEL) Field Descriptions**

Bit	Field	Value	Description
31-1	Reserved	0	Reads return zeros, writes have no effect.
0	ADEVT PSEL		ADEVT Pin Pull Control Select. This bit selects a pull-down or pull-up on the ADEVT pin if it is configured to be an input pin. Any operating mode read/write:
		0	Pull down is selected on ADEVT pin.
		1	Pull up is selected on ADEVT pin.

### 18.11.51 ADC Event Group Sample Cap Discharge Control Register (ADEVSAMPDISEN)

ADC Event Group Sample Cap Discharge Control Register (ADEVSAMPDISEN) is shown in [Figure 18-67](#) and described in [Table 18-54](#).

**Figure 18-67. ADC Event Group Sample Cap Discharge Control Register (ADEVSAMPDISEN)**



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

**Table 18-54. ADC Event Group Sample Cap Discharge Control Register (ADEVSAMPDISEN) Field Descriptions**

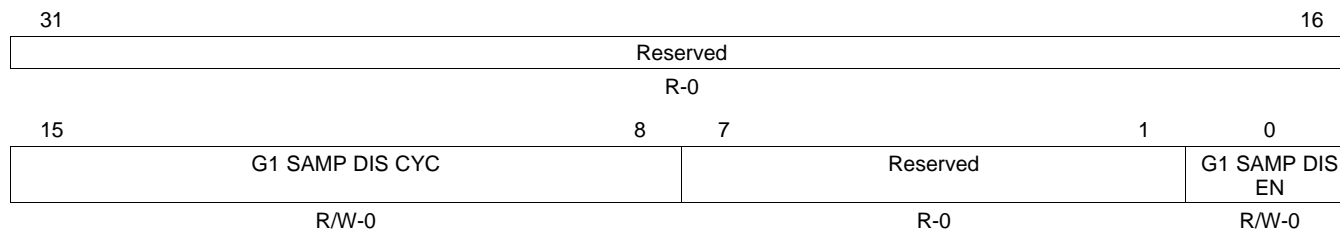
Bit	Field	Value	Description
31-16	Reserved	0	Reads return zeros, writes have no effect.
15-8	EV SAMP DIS CYC		Event Group sample cap discharge cycles. These bits specify the duration in terms of ADCLK cycles for which the ADC internal sampling capacitor is allowed to discharge before sampling the input channel voltage.
7-1	Reserved	0	Reads return zeros, writes have no effect.
0	EV SAMP DIS EN		Event Group sample cap discharge enable. Any operation 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.



### 18.11.52 ADC Group1 Sample Cap Discharge Control Register (ADG1SAMPDISEN)

ADC Group1 Sample Cap Discharge Control Register (ADG1SAMPDISEN) is shown in [Figure 18-68](#) and described in [Table 18-55](#).

**Figure 18-68. ADC Group1 Sample Cap Discharge Control Register (ADG1SAMPDISEN)**



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

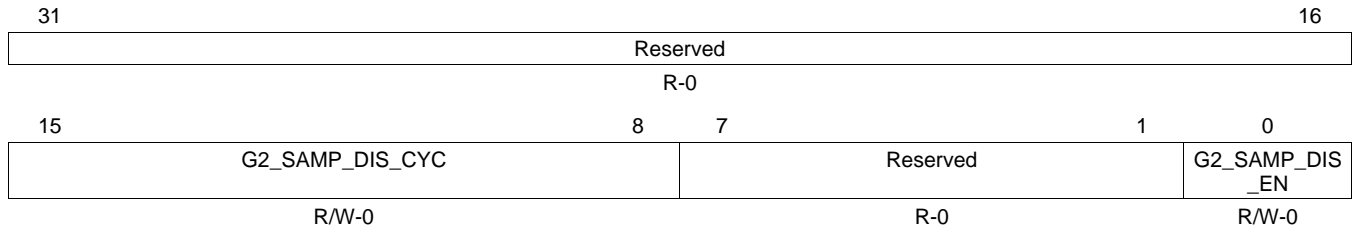
**Table 18-55. ADC Group1 Sample Cap Discharge Control Register (ADG1SAMPDISEN) Field Descriptions**

Bit	Field	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 duration in terms of ADCLK cycles for which the ADC internal sampling capacitor is allowed to discharge before sampling the input channel voltage.
7-1	Reserved	0	Reads return zeros, writes have no effect.
0	G1 SAMP DIS EN		Group1 sample cap discharge enable. Any operation 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.

**18.11.53 ADC Group2 Sample Cap Discharge Control Register (ADG2SAMPDISEN)**

ADC Group2 Sample Cap Discharge Control Register (ADG2SAMPDISEN) is shown in [Figure 18-69](#) and described in [Table 18-56](#).

**Figure 18-69. ADC Group2 Sample Cap Discharge Control Register (ADG2SAMPDISEN)**



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

**Table 18-56. ADC Group2 Sample Cap Discharge Control Register (ADG2SAMPDISEN) Field Descriptions**

Bit	Field	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 duration in terms of ADCLK cycles for which the ADC internal sampling capacitor is allowed to discharge before sampling the input channel voltage.
7-1	Reserved	0	Reads return zeros, writes have no effect.
0	G2 SAMP DIS EN		Group2 sample cap discharge enable. Any operation 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.

### 18.11.54 ADC Magnitude Compare Interrupt Control Registers (ADMAGINTxCR)

Figure 18-70 and Table 18-57 describe the ADMAGINTxCR registers. The ADC module supports up to three magnitude compare interrupts. These registers are at offset addresses 128h, 130h, and 138h.

**Figure 18-70. ADC Magnitude Compare Interrupt Control Registers (ADMAGINTxCR)**

31	30	26	25	16
Reserved	MAG CHIDx			MAG THRx
R-0	R/W-0			R/W-0
15	13	12		
Reserved		COMP CHIDx		
R-0				
7	Reserved		2	1
			CHN/THR COMPx	0
			R/W-0	CMP GE/LTx
			R-0	R/W-0

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

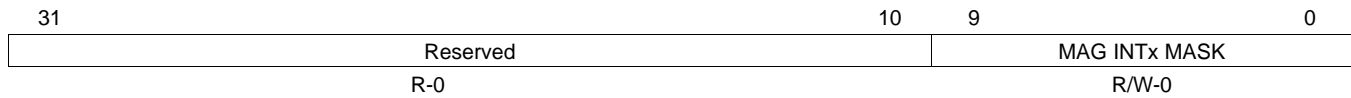
**Table 18-57. ADC Magnitude Compare Interrupt Control Registers (ADMAGINTxCR) Field Descriptions**

Bit	Field	Value	Description
31	Reserved	0	Reads return zeros, writes have no effect.
30-26	MAG CHIDx		These bits specify the channel number from 0 to 31 for which the conversion result needs to be monitored by the ADC.
25-16	MAG THRx		These bits specify the 10-bit compare value which the ADC will use for the comparison with the MAG CHIDx channel's conversion result.
15-16	Reserved	0	Reads return zeros, writes have no effect.
12-8	COMP CHIDx		These bits specify the channel number from 0 to 31 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.  Any operation 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.
0	CMP GE/LTx		"Greater than or equal to" OR "Less than" comparison operator.  Any operation 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).

**18.11.55 ADC Magnitude Compare Interruptx Mask Register (ADMAGINTxMASK)**

Figure 18-71 and described in Table 18-58 describe the ADMAGINTxMASK registers. There are three such registers for the three magnitude compare interrupts. These are located at address offsets 12Ch, 134h, and 13Ch.

**Figure 18-71. ADC Magnitude Compare Mask Register (ADMAGINTxMASK)**



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

**Table 18-58. ADC Magnitude Compare Mask Register (ADMAGINTxMASK) Field Descriptions**

Bit	Field	Value	Description
31-10	Reserved	0	Reads return zeros, writes have no effect.
9-0	MAG INTx MASK	0	These bits specify the mask for the comparison in order to generate the magnitude compare interrupt # x. Any operation mode read/write: The ADC module will not mask the corresponding bit for the comparison.
		1	The ADC module will mask the corresponding bit for the comparison.

### 18.11.56 ADC Magnitude Compare Interrupt Enable Set Register (ADMAGINTENASET)

ADC Magnitude Compare Interrupt Enable Set Register (ADMAGINTENASET) is shown in [Figure 18-72](#) and described in [Table 18-59](#).

**Figure 18-72. ADC Magnitude Compare Interrupt Enable Set Register (ADMAGINTENASET)**

31	Reserved	3	2	0
	R-0		MAG INT ENA SET R/W-0	

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

**Table 18-59. ADC Magnitude Compare Interrupt Enable Set Register (ADMAGINTENASET) Field Descriptions**

Bit	Field	Value	Description
31-3	Reserved	0	Reads return zeros, writes have no effect.
2-0	MAG INT ENA SET		Each of these three bits, when set, enable the corresponding magnitude compare interrupt.
			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.

**18.11.57 ADC Magnitude Compare Interrupt Enable Clear Register (ADMAGINTENACLR)**

ADC Magnitude Compare Interrupt Enable Clear Register (ADMAGINTENACLR) is shown in [Figure 18-73](#) and described in [Table 18-60](#).

**Figure 18-73. ADC Magnitude Compare Interrupt Enable Clear Register (ADMAGINTENACLR)**

31	Reserved	3	2	0
	R-0		MAG INT ENA CLR	
			R/W-0	

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

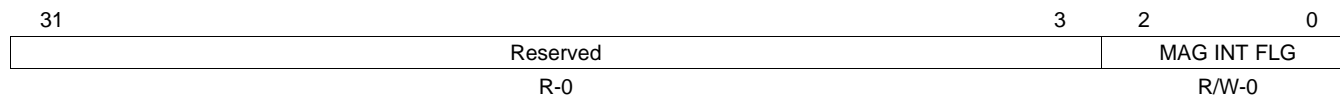
**Table 18-60. ADC Magnitude Compare Interrupt Enable Clear Register (ADMAGINTENACLR) Field Descriptions**

Bit	Field	Value	Description
31-3	Reserved	0	Reads return zeros, writes have no effect.
2-0	MAG INT ENA CLR	0	Each of these three bits, when set, enable the corresponding magnitude compare interrupt.  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.
		1	

### 18.11.58 ADC Magnitude Compare Interrupt Flag Register (ADMAGINTFLG)

ADC Magnitude Compare Interrupt Flag Register (ADMAGINTFLG) is shown in [Figure 18-74](#) and described in [Table 18-61](#).

**Figure 18-74. ADC Magnitude Compare Interrupt Flag Register (ADMAGINTFLG)**



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

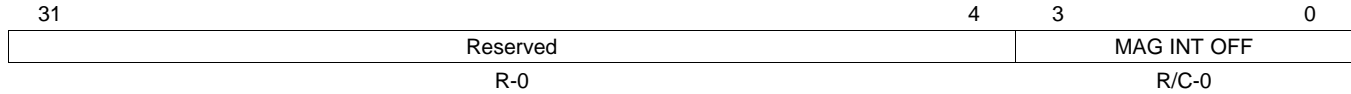
**Table 18-61. ADC Magnitude Compare Interrupt Flag Register (ADMAGINTFLG) Field Descriptions**

Bit	Field	Value	Description
31-3	Reserved	0	Reads return zeros, writes have no effect.
2-0	MAG INT FLG	0	<p>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.</p> <p>Any operation mode, for each bit:</p> <p>Read: The condition for the corresponding magnitude threshold interrupt was false. Write: The corresponding flag is left unchanged.</p>
		1	<p>Read: The condition for the corresponding magnitude threshold interrupt was true. Write: The corresponding flag is cleared. The flag can also be cleared by reading from the magnitude compare interrupt offset register.</p>

### 18.11.59 ADC Magnitude Compare Interrupt Offset Register (ADMAGINTOFF)

ADC Magnitude Compare Interrupt Offset Register (ADMAGINTOFF) is shown in [Figure 18-75](#) and described in [Table 18-62](#).

**Figure 18-75. ADC Magnitude Compare Interrupt Offset Register (ADMAGINTOFF)**



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

**Table 18-62. ADC Magnitude Compare Interrupt Offset Register (ADMAGINTOFF) Field Descriptions**

Bit	Field	Value	Description										
31-4	Reserved	0	Reads return zeros, writes have no effect.										
3-0	MAG INT OFF		<p>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.</p> <p>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.</p> <p>Any operation mode read:</p> <table style="width: 100%; border: none;"> <tr> <td style="padding-left: 20px;">0</td> <td>No magnitude compare interrupt is pending.</td> </tr> <tr> <td style="padding-left: 20px;">1h</td> <td>Magnitude compare interrupt # 1 is pending.</td> </tr> <tr> <td style="padding-left: 20px;">2h</td> <td>Magnitude compare interrupt # 2 is pending.</td> </tr> <tr> <td style="padding-left: 20px;">3h</td> <td>Magnitude compare interrupt # 3 is pending.</td> </tr> <tr> <td style="padding-left: 20px;">4h-Fh</td> <td>Reserved. These combinations do not occur.</td> </tr> </table>	0	No magnitude compare interrupt is pending.	1h	Magnitude compare interrupt # 1 is pending.	2h	Magnitude compare interrupt # 2 is pending.	3h	Magnitude compare interrupt # 3 is pending.	4h-Fh	Reserved. These combinations do not occur.
0	No magnitude compare interrupt is pending.												
1h	Magnitude compare interrupt # 1 is pending.												
2h	Magnitude compare interrupt # 2 is pending.												
3h	Magnitude compare interrupt # 3 is pending.												
4h-Fh	Reserved. These combinations do not occur.												



### 18.11.60 ADC Event Group FIFO Reset Control Register (ADEVFIFORESETCR)

ADC Event Group FIFO Reset Control Register (ADEVFIFORESETCR) is shown in [Figure 18-76](#) and described in [Table 18-63](#).

**Figure 18-76. ADC Event Group FIFO Reset Control Register (ADEVFIFORESETCR)**

31	Reserved	1	0
	R-0		EV FIFO RESET R/W-0

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

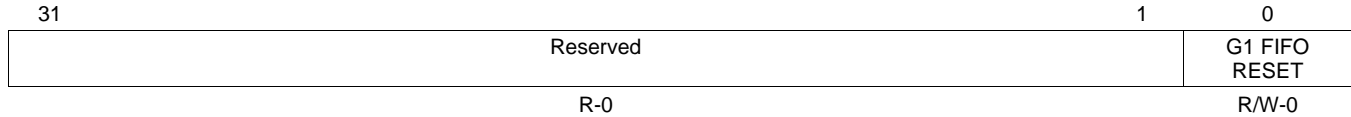
**Table 18-63. ADC Event Group FIFO Reset Control Register (ADEVFIFORESETCR) Field Descriptions**

Bit	Field	Value	Description
31-1	Reserved	0	Reads return zeros, writes have no effect.
0	EV FIFO RESET		<p>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.</p> <p>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'.</p> <p>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.</p> <p>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'.</p>

**18.11.61 ADC Group1 FIFO Reset Control Register (ADG1FIFORESETCR)**

ADC Group1 FIFO Reset Control Register (ADG1FIFORESETCR) is shown in [Figure 18-77](#) and described in [Table 18-64](#).

**Figure 18-77. ADC Group1 FIFO Reset Control Register (ADG1FIFORESETCR)**



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

**Table 18-64. ADC Group1 FIFO Reset Control Register (ADG1FIFORESETCR) Field Descriptions**

Bit	Field	Value	Description
31-1	Reserved	0	Reads return zeros, writes have no effect.
0	G1 FIFO RESET	0-1	<p>ADC Group1 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 Group1 results memory starting from the first location.</p> <p>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'.</p> <p>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.</p> <p>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'.</p>

### 18.11.62 ADC Group2 FIFO Reset Control Register (ADG2FIFORESETCR)

ADC Group2 FIFO Reset Control Register (ADG2FIFORESETCR) is shown in [Figure 18-78](#) and described in [Table 18-65](#).

**Figure 18-78. ADC Group2 FIFO Reset Control Register (ADG2FIFORESETCR)**

31	Reserved	1	0
			G2 FIFO RESET
R-0			R/W-0

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

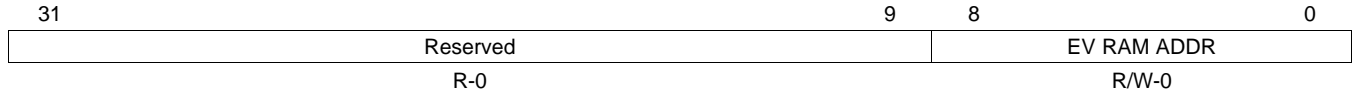
**Table 18-65. ADC Group2 FIFO Reset Control Register (ADG2FIFORESETCR) Field Descriptions**

Bit	Field	Value	Description
31-1	Reserved	0	Reads return zeros, writes have no effect.
0	G2 FIFO RESET	0-1	<p>ADC Group2 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 Group2 results memory starting from the first location.</p> <p>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'.</p> <p>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.</p> <p>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'.</p>

**18.11.63 ADC Event Group RAM Write Address Register (ADEVAMWRADDR)**

ADC Event Group RAM Write Address Register (ADEVAMWRADDR) is shown in [Figure 18-79](#) and described in [Table 18-66](#).

**Figure 18-79. ADC Event Group RAM Write Address Register (ADEVAMWRADDR)**



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

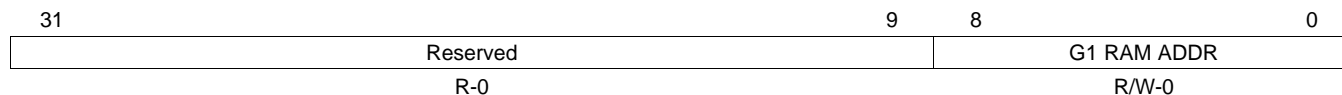
**Table 18-66. ADC Event Group RAM Write Address Register (ADEVAMWRADDR) Field Descriptions**

Bit	Field	Value	Description
31-9	Reserved	0	Reads return zeros, writes have no effect.
8-0	EV RAM ADDR		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 until that time.

### 18.11.64 ADC Group1 RAM Write Address Register (ADG1RAMWRADDR)

ADC Group1 RAM Write Address Register (ADG1RAMWRADDR) is shown in [Figure 18-80](#) and described in [Table 18-67](#).

**Figure 18-80. ADC Group1 RAM Write Address Register (ADG1RAMWRADDR)**



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

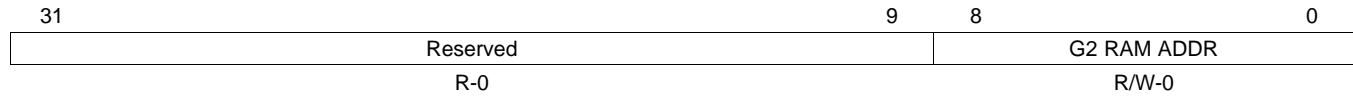
**Table 18-67. ADC Group1 RAM Write Address Register (ADG1RAMWRADDR) Field Descriptions**

Bit	Field	Value	Description
31-9	Reserved	0	Reads return zeros, writes have no effect.
8-0	G1 RAM ADDR		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 until that time.

### 18.11.65 ADC Group2 RAM Write Address Register (ADG2RAMWRADDR)

ADC Group2 RAM Write Address Register (ADG2RAMWRADDR) is shown in [Figure 18-81](#) and described in [Table 18-68](#).

**Figure 18-81. ADC Group2 RAM Write Address Register (ADG2RAMWRADDR)**



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

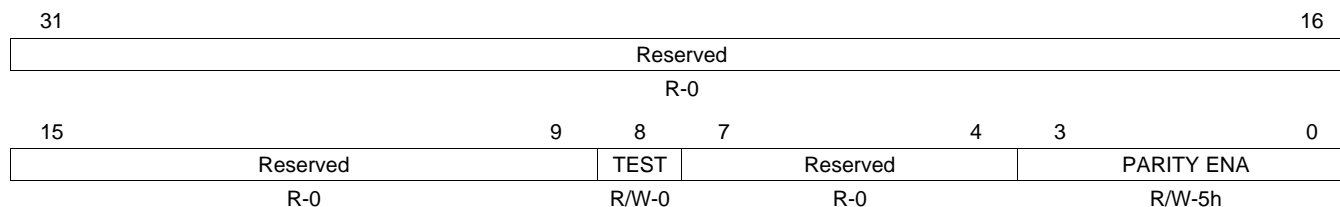
**Table 18-68. ADC Group2 RAM Write Address Register (ADG2RAMWRADDR) Field Descriptions**

Bit	Field	Value	Description
31-9	Reserved	0	Reads return zeros, writes have no effect.
8-0	G2 RAM ADDR		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 until that time.

### 18.11.66 ADC Parity Control Register (ADPARCR)

ADC Parity Control Register (ADPARCR) is shown in [Figure 18-82](#) and described in [Table 18-69](#).

**Figure 18-82. ADC Parity Control Register (ADPARCR)**



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

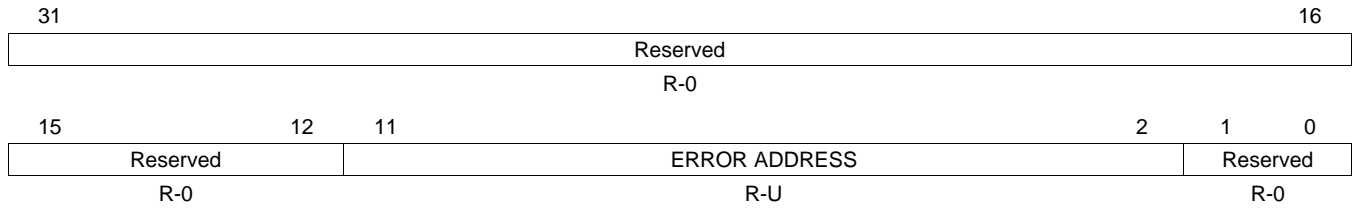
**Table 18-69. ADC Parity Control Register (ADPARCR) Field Descriptions**

Bit	Field	Value	Description
31-9	Reserved	0	Reads return zeros, writes have no effect.
8	TEST	0 1	<p>This bit maps the parity bits into the ADC results' RAM frame so that the application can access them.</p> <p>Any operation mode read, privileged mode write:</p> <p>0 The parity bits are not memory-mapped.</p> <p>1 The parity bits are memory mapped.</p>
7-4	Reserved	0	Reads return zeros, writes have no effect.
3-0	PARITY ENA	5h Any other	<p>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.</p> <p>If parity checking is enabled and a parity error is detected the ADC module sends a parity error signal to the System module.</p> <p>Any operation mode read, privileged mode write:</p> <p>5h Parity check is disabled.</p> <p>Any other Parity check is enabled.</p>

**18.11.67 ADC Parity Error Address Register (ADPARADDR)**

ADC Parity Error Address Register (ADPARADDR) is shown in [Figure 18-83](#) and described in [Table 18-70](#).

**Figure 18-83. ADC Parity Error Address Register (ADPARADDR)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset; -U = value after reset is unknown

**Table 18-70. ADC Parity Error Address Register (ADPARADDR) Field Descriptions**

Bit	Field	Value	Description
31-12	Reserved	0	Reads return zeros, writes have no effect.
11-2	ERROR ADDRESS		These bits hold the address of the first parity error generated in the ADC results' RAM. This error address is frozen from being updated until it is read by the application. In emulation mode, this address is maintained frozen even when read.
1-0	Reserved	0	Reads return zeros, writes have no effect. Reading [11:0] provides the 32-bit aligned address.



### 18.11.68 ADC Power-Up Delay Control Register (ADPWRUPDLYCTRL)

Figure 18-84 and Table 18-71 describe the ADPWRDLYCTRL register.

**Figure 18-84. ADC Power-Up Delay Control Register (ADPWRUPDLYCTRL)**

31	Reserved	10	9	0
	R-0			PWRUP DLY R/W-0

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

**Table 18-71. ADC Power-Up Delay Control Register (ADPWRUPDLYCTRL) Field Descriptions**

Bit	Field	Value	Description
31-10	Reserved	0	Reads return zeros, writes have no effect.
9-0	PWRUP DLY		This register defines the number of VCLK cycles that the ADC state machine has to wait after releasing the ADC core from power down before starting a new conversion. Please refer to <a href="#">Section 18.8.3</a> , ADC Power-Down Mode for more details.

## **High-End Timer (N2HET) Module**

This chapter provides a general description of the High-End Timer (N2HET). The N2HET is a software-controlled timer with a dedicated specialized timer micromachine and a set of 30 instructions. The N2HET micromachine is connected to a port of up to 32 input/output (I/O) pins.

Topic	Page
<b>19.1 Features</b> .....	<b>904</b>
<b>19.2 N2HET Functional Description</b> .....	<b>908</b>
<b>19.3 Angle Functions</b> .....	<b>937</b>
<b>19.4 N2HET Control Registers</b> .....	<b>941</b>
<b>19.5 Instruction Set</b> .....	<b>976</b>

## 19.1 Features

- Programmable timer for input and output timing functions
- Reduced instruction set (30 instructions) for dedicated time and angle functions
- Up to 160 96-bit words of instruction RAM protected by parity
- User defined configuration of 25-bit virtual counters for timer, event counters and angle counters
- 7-bit hardware counters for each pin allow up to 32-bit resolution in conjunction with the 25-bit virtual counters
- Up to 32 pins usable for input signal measurements or output signal generation
- Programmable suppression filter for each input pin with adjustable suppression window
- Low CPU overhead and interrupt load
- Efficient data transfer to or from the CPU memory with dedicated High-End-Timer Transfer Unit (HTU) or DMA
- Diagnostic capabilities with different loopback mechanisms and pin status readback functionality
- Hardware Angle Generator (HWAG) - (More information will be available in future User's Guide Revision)

### 19.1.1 Overview

The N2HET is a fifth-generation Texas Instruments (TI) advanced intelligent timer module. It provides an enhanced feature set compared to previous generations.

This timer module provides sophisticated timing functions for real-time applications such as engine management or motor control. The high resolution hardware channels allow greater accuracy for widely used timing functions such as period and pulse measurements, output compare, and PWMs.

The reduced instruction set, based mostly on very simple, but comprehensive instructions, improves the definition and development cycle time of an application and its derivatives. The N2HET breakpoint feature, combined with various stop capabilities, makes the N2HET software application easy to debug.

#### 19.1.1.1 Major Advantages

In addition to classic time functions such as input capture or multiple PWMs, higher-level time functions can be easily implemented in the timer program main loop. Higher-level time functions include angle driven wave forms, angle- and time-driven pulses, and input pulse width modulation (PWM) duty cycle measurement.

Because of these high-level functions, data exchanges with the CPU are limited to the fundamental parameters of the application (periods, pulse widths, angle values, etc.); and the real-time constraints for parameter communication are dramatically minimized; for example, few interrupts are required and asynchronous parameter updates are allowed.

The reduced instruction set and simple execution flow control make it simple and easy to develop and modify programs. Simple algorithms can embed the entire flow control inside the N2HET program itself. More complex algorithms can take advantage of the CPU access to the N2HET RAM. With this, the CPU program can make calculations and can modify the timer program flow by changing the data and control fields of the N2HET RAM. CPU access to the N2HET RAM also improves the debug and development of timer programs. The CPU program can stop the N2HET and view the contents of the program, control, and data fields that reside in the N2HET RAM.

Finally, the modular structure provides maximum flexibility to address a wide range of applications. The timer resolution can be selected from two cascaded prescalers to adjust the loop resolution and HR clocks. The 32 I/O pins can provide any combination of input, period or pulse capture, and output compare, including high resolution for each channel.

#### 19.1.1.2 Timer Module Structure and Execution

The timer consists of a specialized micromachine that operates a reduced instruction set. Two 25-bit

registers and three 32-bit registers are available to manipulate information such as time, event counts, and angle values. System performance is improved by a wide instruction format (96 bits) that allows the N2HET to fetch the instructional operation code and data in one system cycle, thus increasing the speed at which data can be processed. The typical operations performed in the ALU are additions (count), compares, and magnitude compares (higher or same).

Each instruction is made up of a 32-bit program field, a 32-bit control field and a 32-bit data field. The N2HET execution unit fetches the complete 96-bit instruction in one cycle and executes it. All instructions include a 9-bit field for specifying the address of the next instruction to be executed. Some instructions also include a 9-bit conditional address, which is used as the next address whenever a particular condition is true. This makes controlling the flow of an N2HET program inexpensive; in many cases a separate branch instruction is not required.

The interface to the host CPU is based on both communication memory and control registers. The communication memory includes timer instructions (program and data). This memory is typically initialized by the CPU or DMA after reset before the timer starts execution. Once the timer program is loaded into the memory, the CPU starts the timer execution, and typically data parameters are then read or written into the timer memory. The control registers include bits for selecting timer clock, configuring I/O pins, and controlling the timer module.

The programmer implements timer functions by combining instructions in specific sequences. For instance, a single count (CNT) instruction implements a timer. A simple PWM generator can be implemented with a two instruction sequence: CNT and compare (ECMP or MCMP). A complex time function may include many instructions in the sequence. The total timer program is a set of instructions executed sequentially, one after the other. Reaching the end, the program must roll to the first instruction so that it behaves as a loop. The time for a loop to execute is referred to as a *loop resolution clock cycle* or *loop resolution period (LRP)*. When the N2HET rolls over to the first instruction, the timer waits for the resolution clock to restart the execution of the loop to ensure that only one loop is executed for each loop resolution clock.

The longest path through an N2HET program must be completed within the loop resolution clock (LRP). Otherwise, the program will execute unpredictably because some instructions will not be executed each time through the loop. This effect creates a strong link between the accuracy of the timer functions and the number of functions (the number of instructions) the timer can perform. High resolution (HR) hardware timer extensions are available for each of the N2HET pins to help overcome this limitation.

The high resolution hardware timers operate from the *high resolution clock*, which may be configured for frequency multiples between 2 and 128 times the loop resolution clock frequency. This extending the resolution of timer events and measurements well beyond what is possible with only loop resolution instructions.

Most of the commonly used N2HET instructions can operate either at loop resolution or high resolution; with the restriction that for each pin at most one high resolution instruction can be executed per loop resolution period.

Certain instructions (MOV32, ADM32, ...) allow the manipulation of certain fields of other instructions. These built-in move instructions actually transfer new data 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 when the CPU updates the data asynchronously.

### 19.1.1.3 Performance

Most instructions execute in one cycle, but a few take two or three cycles.

The N2HET can generate many complex output waveforms without CPU interrupts. Where special algorithms are needed following a specific event (e.g., missing teeth or a short/long input signal), a minimal number of interrupts to the CPU are needed freeing the CPU for other tasks.

### 19.1.1.4 N2HET Compared to NHET

N2HET Enhancements from NHET include:

- Eight new instructions: ADD, ADC, SUB, SBB, AND, OR, XOR, RCNT

- Full set of ALU flags Carry (C), Negative (N), Zero (Z), Overflow (V)
- Branch instruction (BR) extended to support signed and unsigned arithmetic comparison conditions
- Two additional 32-bit temporary working registers R, S.
- New HETAND register for AND-Sharing of High Resolution structure between pairs of pins
- Improved high resolution PCNT instruction

#### 19.1.1.5 NHET and N2HET Compared to HET

Compared to the HET module, the N2HET contains all of the enhancements described in [Section 19.1.1.4](#) plus the following additional enhancements:

- New Interrupt Enable Set and Clear registers
- Capability to generate requests to the DMA module or the HET Transfer Unit (HTU) including new Request Enable Set and Clear registers
- NHET RAM parity error detection
- Suppression filters for each of the 32 I/O channel and control register to configure the limiting frequency and counter clock
- Enhanced edge detection hardware that does not rely on the previous bit field in the control word of the NHET instruction.
- The next, conditional and remote addresses are extended from 8 to 9 bits
- The loop resolution data fields are extended from 20 to 25 bits
- The high resolution data fields are extended from 5 to 7 bits
- Instructions with an adequate condition are able to specify the number of the request line, which triggers either the HET Transfer Unit (HTU) or the DMA module
- The CNT instruction provides a bit, which allows to configure either an equal comparison or a greater or equal comparison when comparing the selected register value with the Max-value
- The MOV32 instruction provides a new bit. If set to one the MOV32 will only perform the move, when the Z-flag is set. If set to zero the MOV32 will perform the move whenever it is executed (independent on the state of the Z-flag)
- There is a new instruction WCAPE, which is a combination of a time stamp and an edge counter
- New Open Drain, Pull Disable, and Pull Select registers

#### 19.1.1.6 Instructions Features

The N2HET has the following instructions features:

- N2HET uses a RISC-based specialized timer micromachine to carry out a set of 30 instructions
- Instructions are implemented in a Very Long Instruction Word (VLIW) format (96 bits wide)
- The N2HET program execution is self-driven by external or internal events, branching to special routines based on input edges or output compares
- Instructions point to the next instruction executed, eliminating the need for a program counter
- Several instructions can change the program flow based on internal or external conditions

#### 19.1.1.7 Program Usage

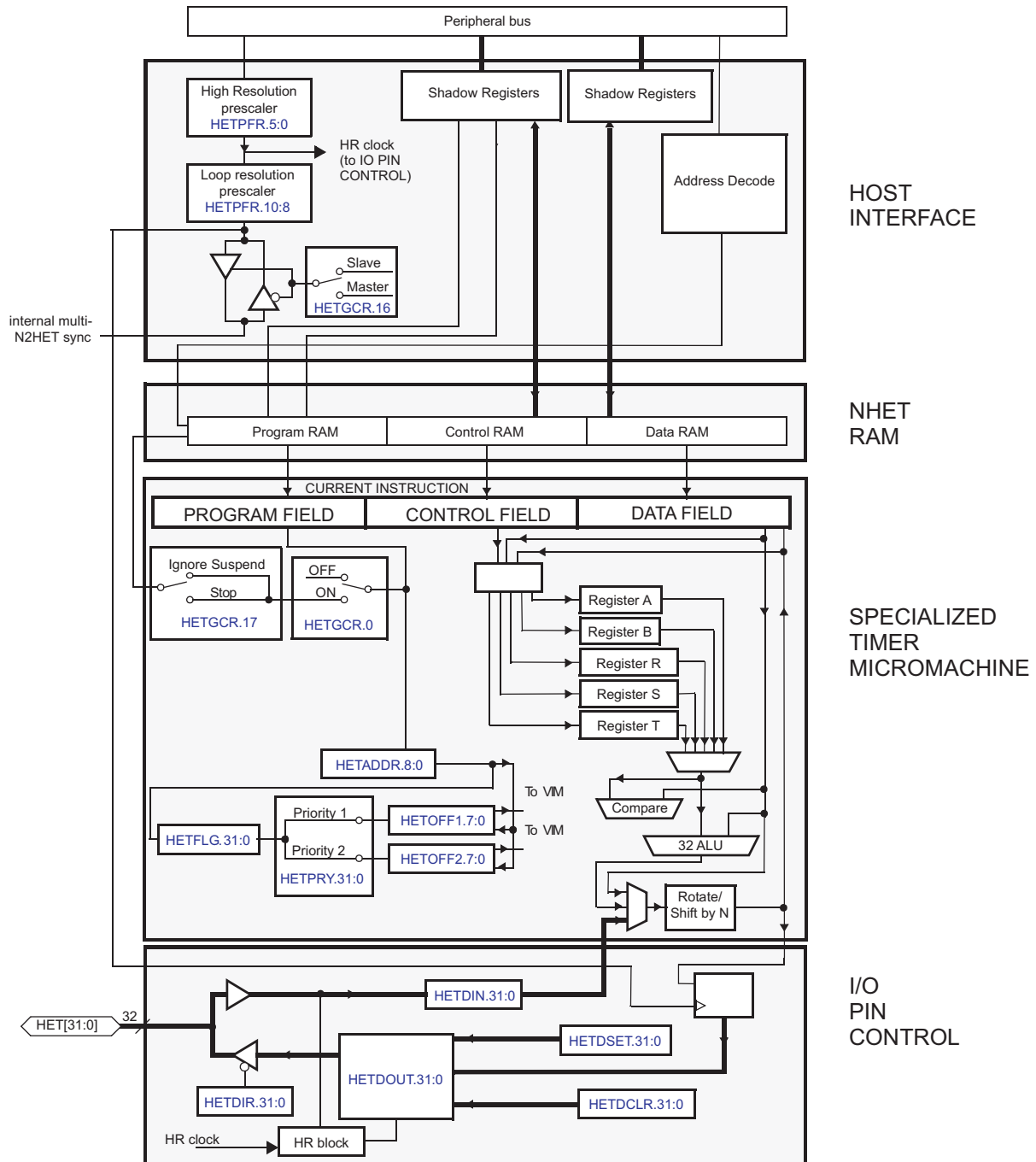
The N2HET instructions/program can be assembled with the N2HET assembler. The assembler generates a C-structure which can be included into the main application program. The application has to copy the content of the structure into the N2HET RAM, set up necessary registers and start the N2HET program execution. In addition to the C-structure, the assembler generates also a header file which makes it easy for the main application to access the different instructions and change for example the duty cycle of a PWM or read out the captured value of a specific signal edge.

### 19.1.2 Block Diagram

The N2HET module (see [Figure 19-1](#)) comprises four separate components:

- Host interface
- N2HET RAM
- Specialized timer micromachine
- I/O control (the N2HET is attached to an I/O port of up to 32 pins)

Figure 19-1. N2HET Block Diagram



## 19.2 N2HET Functional Description

The N2HET contains RAM into which N2HET code is loaded. The N2HET code is run by the specialized timer micromachine. The host interface and I/O control provide an interface to the CPU and external pins respectively.

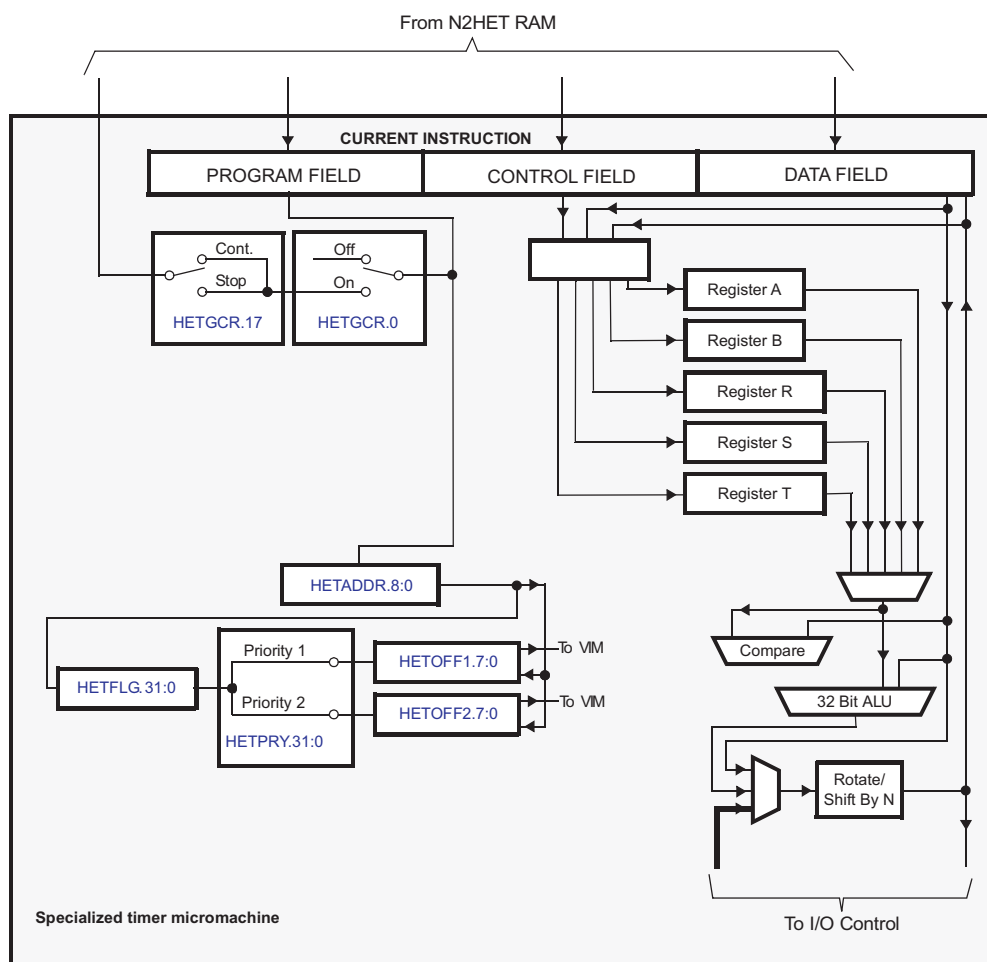
### 19.2.1 Specialized Timer Micromachine

The N2HET has its own instruction set, detailed in [Section 19.5.1](#). The timer micromachine reads each instruction from the N2HET RAM. The program and control fields contain the instructions for how the specialized timer micromachine executes the command. For most instructions, the data field stores the information that needs to be manipulated.

The specialized timer micromachine executes the instructions stored in the N2HET RAM sequentially. The N2HET program execution is self-driven by external or internal events. This means that input edges or output compares may force the program to branch to special routines using a conditional address.

[Figure 19-2](#) shows some of the major operations that the N2HET can carry out, namely compares, captures, angle functions, additions, and shifts. The N2HET contains five registers (A, B, R, S, and T) used to hold compare or counter values and are used by the N2HET instructions. Data may be taken from the registers or the data field for manipulation; likewise, the data may be returned to the registers or the data field.

**Figure 19-2. Specialized Timer Micromachine**



### 19.2.1.1 Time Slots and Resolution Loop

Each instruction requires a specific number of cycles or time slots to execute. The resolution specified in the prescaler bitfields determines the timer accuracy. All input captures, event counts, and output compares are executed once in each resolution loop. HR captures and compares are possible (up to N2HET clock accuracy) on the HR I/O pins. For more information about the HR I/O structure, see [Section 19.2.5](#).

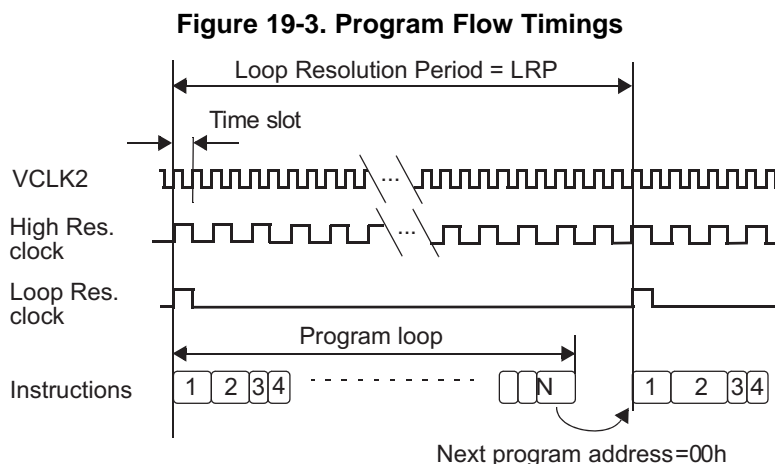
### 19.2.1.2 Program Loop Time

The program loop time is the sum of all cycles used for instruction execution. This time may vary from one loop to another if the N2HET program includes conditionally executed instructions.

The timer program restarts on every resolution loop. The start address is fixed at N2HET RAM address 00h. The longest path through a program must fit within one loop resolution period to guarantee complete accuracy.

The last instruction of a program must branch back to the fixed start address (next program address=00h). When an N2HET program branches back to address 00h before the end of a loop resolution period, the N2HET detects this and pauses instruction execution until the beginning of the next loop resolution period.

The timing diagram below illustrates the program flow execution.



### 19.2.1.3 Instruction Execution Sequence

The execution of a N2HET program begins with the first occurrence of the loop resolution clock, after the N2HET is turned on. At the first and subsequent occurrences of the loop resolution, the instruction at location address 00h is prefetched. The program execution begins at the occurrence of the loop resolution clock and continues executing the instructions until the program branches to 00h location. The instruction is prefetched at location 00h and execution flag is reset. The N2HET pauses instruction execution until the occurrence of the loop resolution clock and resumes normal execution.

N2HET programs must be written so that they complete execution and return to address 00h before the occurrence of the next loop resolution clock. If the N2HET program exceeds this execution time limit, then a program overflow condition occurs as described in [Section 19.2.1.4](#).

### 19.2.1.4 Program Overflow Condition

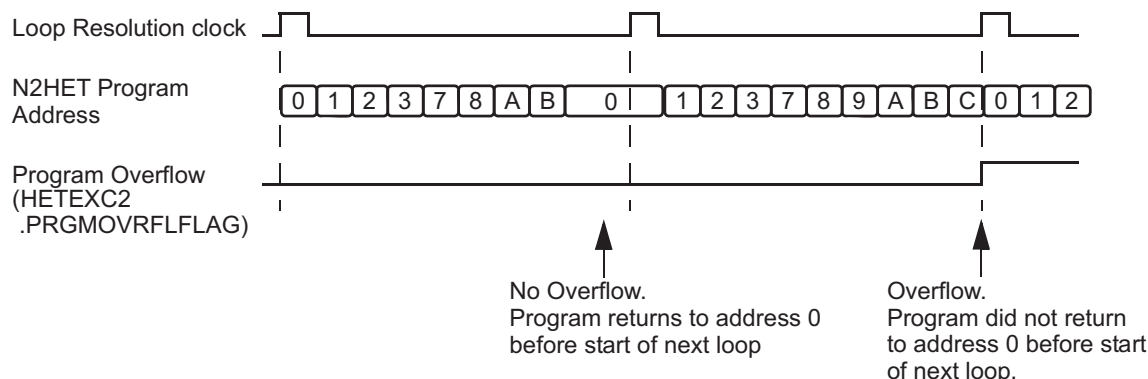
If the number of time slots used in a program loop exceeds the number available time slots in one loop resolution, the timer sets the program overflow interrupt flag located in the HETEXC2 register. To maintain synchronization of the I/Os, this condition should never be allowed to occur in a normal operation. The HETEXC2.PRGMOVRFLFLAG flag provides a mechanism for checking that the condition does not occur during the debug and validation phases.



As [Figure 19-4](#) illustrates, when a program overflow occurs, the currently executing N2HET program sequence is interrupted and restarted at N2HET address 0 for the beginning of the next loop resolution clock period. Also, HETEXC2.PRGMOVRFLFLAG is set.

If the instruction that caused the overflow (instruction at address 0xC in [Figure 19-4](#)) has any pin actions selected, these pin actions will not be performed. However other actions of the instruction including register and RAM updates will still be performed.

**Figure 19-4. Use of the Overflow Interrupt Flag (HETEXC2)**



### 19.2.1.5 Architectural Restrictions on N2HET Programs

Certain architectural restrictions apply to N2HET programs:

1. The size of an N2HET program must be greater than one instruction.
2. An extra wait state is incurred by any instruction that modifies a field in the next instruction to be executed.
3. Only one instruction (using high resolution) is allowed per high resolution pin.
4. Consecutive break points are not supported. Instructions with break points must have at least a distance of two instructions (e.g. at addresses 1, 3, 5, 7, etc. assuming the program executes linearly)

---

**NOTE:** While it would be unusual to code an N2HET program that is only one instruction long, it is trivial to modify such a program to meet the requirement of restriction 1. Simply add a second instruction to the program, which may be a simple branch to zero.

---



---

**NOTE:** To enforce restriction 3, the high resolution pin structures respond only to the first instruction that is executed matching their pin number with hr\_lr=HIGH, regardless of whether or not the en\_pin\_action field is ON. Subsequent instructions are ignored by the high resolution pin structure for the remainder of the loop resolution period.

---

### 19.2.1.6 Multi-Resolution Scheme

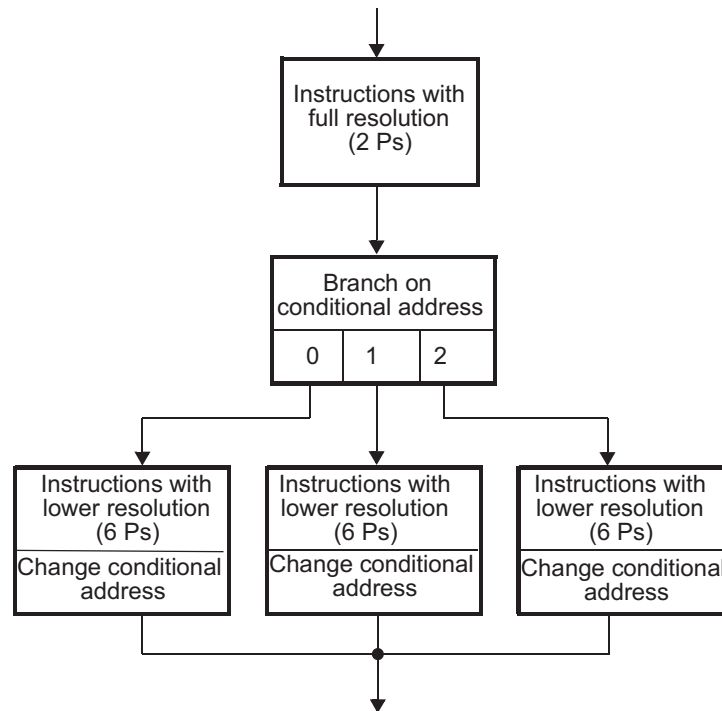
The N2HET has the capability to virtually extend the counter width by executing instructions only once every N loop resolution periods. This decreases the timer resolution, but extends the counter range which may be useful when generating or measuring slow signals. [Figure 19-5](#) illustrates how a multi-resolution scheme may be implemented in an N2HET program. An unconditional Branch instruction and an index sequence, using a MOV64 instruction in each low resolution loop, is required to control this particular program flow.

---

**NOTE:** HR instructions must be placed in the main (full resolution) loop to ensure proper operation.

---

Figure 19-5. Multi-Resolution Operation Flow Example



### 19.2.1.7 Debug Capability

The N2HET supports breakpoints to allow you to more easily debug your N2HET program. [Figure 19-6](#) provides an illustration of the breakpoint mechanism.

The steps to enable an N2HET breakpoint are:

1. Make sure the device nTRST pin is high, since N2HET breakpoints are disabled whenever this pin is low. (Normally this is handled automatically when a JTAG debugger is attached).
2. Attach a JTAG debugger and connect to the TMS570 device that has been already programmed with the N2HET code that needs to be debugged. (downloading to on-chip flash is outside the scope of this section).
3. Execute the CPU program at least until the point where the N2HET program RAM has been initialized by the CPU.
4. Open a memory window in the N2HET registers.
5. Make sure NHETEXC2.DEBUGSTATUSFLAG is cleared.
6. Open a memory window on the N2HET RAM
7. Set bit 22 in the program field of the instruction(s) on which you wish to break. Note that this instruction will be executed **before** the N2HET is halted - slightly different from how CPU breakpoints behave.
8. Make sure the CPU and N2HET are running, if they are halted then restart the CPU through the JTAG emulator (N2HET will start when the CPU starts).
9. Both the CPU and N2HET will halt when breakpoint is reached.

When the N2HET is halted, its state machines are frozen but all of the N2HET control registers can be accessed through the JTAG emulator interface.

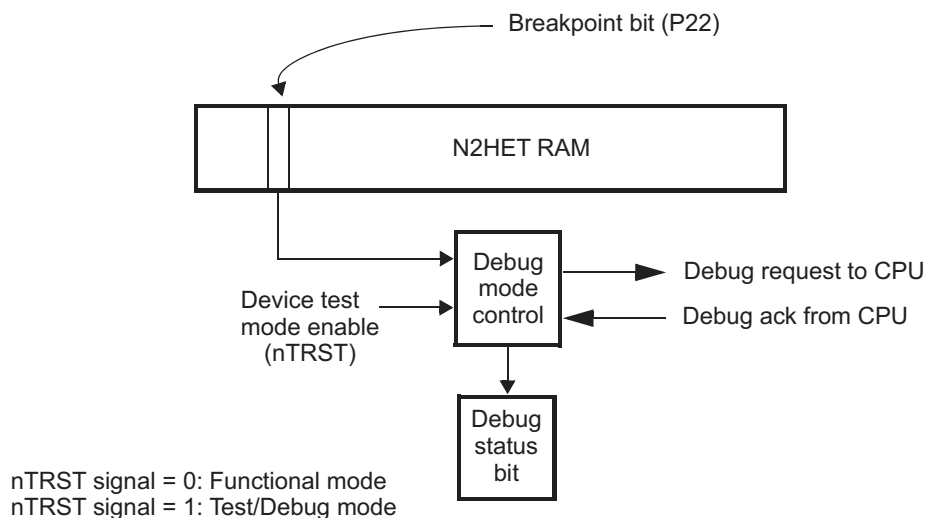
The current N2HET instruction address can be inspected by reading the HETADDR register; this should be pointing to the instruction that caused the breakpoint.

The N2HET internal working registers (A,B,R,S,T) are not directly visible through the JTAG emulator interface. If the content of these registers needs to be inspected, it is best to add an instruction like MOV32 which copies the register value to the N2HET RAM. This RAM location can be inspected when the N2HET halts.

To restart execution of both the CPU and the N2HET from the halted state:

1. Clear NHETEXC2.DEBUGSTATUSFLAG.
2. Clear bit 22 in the program field of the instruction on which the breakpoint was reached.
3. Restart the CPU through the normal JTAG emulator procedure (i.e. 'Run' or 'Go'). The N2HET will automatically start executing when it sees that the CPU has exited the debug state.

**Figure 19-6. Debug Control Configuration**



**NOTE:** Consecutive break points are not supported. Instructions with break points must have at least a distance of two instructions (e.g. at N2HET addresses 1, 3, 5, 7, etc.)

### 19.2.2 N2HET RAM Organization

The N2HET RAM is organized into two sections. The first contains the N2HET program itself. The second contains parity protection bits for the N2HET program.

Each N2HET instruction is 96-bits wide but aligned to a 128-bit boundary. Instructions consist of three 32-bit fields: Program, Control, and Data. Instructions are separated by a fourth unimplemented address to force alignment to 128-bit boundaries.

The integrity of the N2HET program can be protected by Parity. Parity protection is enabled through the N2HET Parity Control Register (HETPCR).

Table 19-1 shows the base addresses for N2HET RAM and N2HET Parity RAM.

**Table 19-1. N2HET RAM Base Addresses**

N2HET1 Base Address	N2HET2 Base Address	Memory
0xFF46_0000	0xFF44_0000	N2HET Instruction RAM (Program/Control/Data)
0xFF46_2000	0xFF44_2000	N2HET Parity RAM

### 19.2.2.1 N2HET RAM Banking

Because the CPU must make updates to the N2HET RAM while the N2HET is executing, for example to update the duty cycle value of a PWM, it is important to understand how the N2HET RAM organization facilitates simultaneous accesses by both the HOST CPU and the N2HET.

The N2HET RAM is implemented as 4 banks of 96-bit wide two port RAM. This means that there a total of 8 ports available; four read and four write. Normally the N2HET will use up two two of these ports at a time. One read port is used to allow the N2HET to prefetch the next N2HET instruction while a write port may be used to update the data or control fields that have changed as a result of executing the current instruction.

N2HET accesses to its own internal RAM are given priority over accesses from an external host (CPU or DMA), this makes N2HET program execution deterministic which is a critical requirement for a timer.

Most N2HET instructions execute in a single cycle. Cases where a wait state impacts the N2HET program execution time are:

- The current N2HET instruction writes data back to the next N2HET in the execution sequence.
- The external host reads from an N2HET instruction where the automatic read-clear option is set, while the N2HET is executing from/on the same address (See [Section 19.2.4.3](#)).

Except for the case of automatic read-clear, the external host is stalled when the host and N2HET have a bank conflict. However this will typically only result in a stall of one cycle, due to the N2HET bank ordering which is organized on the N2HET Address least significant bit boundaries (See [Table 19-2](#)).

Assuming most of the N2HET program executes linearly through the N2HET Address space; if a bank conflict does exist it is usually resolved in the next cycle as the N2HET program moves to the next bank. N2HET programmers should avoid writing a program that accesses the same bank of N2HET RAM on every cycle, as this could lock the external host out of the N2HET memory completely.

[Table 19-2](#) describes the N2HET memory map, as viewed by the N2HET as well as from the memory space of the host CPU and DMA.

**Table 19-2. N2HET RAM Bank Structure**

N2HET Address	Host CPU or DMA Address Space				N2HET RAM Bank
	Program Field Address	Control Field Address	Data Field Address	Reserved Address	
000h	XX0000h	XX0004h	XX0008h	XX000Ch	A
001h	XX0010h	XX0014h	XX0018h	XX001Ch	B
002h	XX0020h	XX0024h	XX0028h	XX002Ch	C
003h	XX0030h	XX0034h	XX0038h	XX003Ch	D
004h	XX0040h	XX0044h	XX0048h	XX004Ch	A
:	:	:	:	:	:
03Fh	XX03F0h	XX03F4h	XX03F8h	XX03FCh	D
040h	XX0400h	XX0404h	XX0408h	XX040Ch	A
:	:	:	:	:	:
1FFh	XX1FF0h	XX1FF4h	XX1FF8h	XX1FFCh	D

---

**NOTE:** The external host interface supports any access size for reads, but only 32-bit writes to the N2HET RAM are supported. Reserved addresses should not be accessed, the result of doing so is indeterminate.

---

### 19.2.2.2 Parity Checking

The N2HET module can detect parity errors in N2HET RAM. As described in [Section 19.2.2](#) the N2HET allows 32-bit writes only. Therefore N2HET RAM parity checking is implemented using one parity bit per 32-bit field in N2HET RAM.

Even or odd parity selection for N2HET parity detection can be configured in the system module. Parity calculation and checking can be enabled/disabled by a 4-bit key in HETPCR.

During a read access to the N2HET RAM, the parity is calculated based on the data read from the RAM and compared with the good parity value stored in the parity bits. The parity check is performed when the N2HET execution unit makes a read access to N2HET RAM, but also when a different master (e.g. CPU, HTU, DMA,...) performs the read access. If any 32-bit-word fails the parity check then an error is signaled to the ESM module. The N2HET address, which generated the error is detected and is captured in HETPAR for host system debugging. The address is frozen from being updated until it is read by the bus master.

The N2HET execution unit reads the instructions, which are 96-bit wide. They contain the program-, control- and data-field whereby each is 32-bit wide. So when fetching N2HET instructions parity checking is performed on three words in parallel.

If a parity error is detected in two or more words in the same cycle then only one address (word at the lower address) is captured. The captured N2HET address is always aligned to a 32-bit word boundary.

During debug, parity checking is still performed on accesses originating from the on-chip host CPU and DMA. However, parity errors that are detected during an access initiated by the debugger itself are ignored.

### 19.2.2.3 Parity Error Detection Actions

Detection of a N2HET parity error causes the following actions:

1. An error is signaled to the ESM module.
2. The Parity Address Register (HETPAR) is loaded with the address of the faulty N2HET field.
3. N2HET execution immediately stops. (The instruction that triggered the parity error is not executed.)
4. The Turn-On/Off-Bit in the N2HET Global Configuration Register (HETGCR) is automatically cleared.
5. All N2HET internal flags are cleared.
6. All N2HET pins selected by N2HET Parity Pin Register (HETPPR) enter a predefined safe state.
7. Register HETDOUT is also updated to reflect changes in pin state due to HETPPR.

The safe state for N2HET pins selected through the HETPPR register depends on how the pin is configured in the HETDIR, HETPDR, and HETPSL registers. [Table 19-3](#) explains how the safe state is determined.

**Table 19-3. Pin Safe State Upon Parity Error Detection**

Safe State	HETDIR	HETPDR	HETPSL
Drive Low	1	0	0
Drive High	1	0	1
High Impedance	1	1	x

### 19.2.2.4 Testing Parity Detection Logic

To test the parity detection logic, the parity RAM has to be made accessible to the CPU in order to allow a diagnostic program to insert parity errors. The control register bit HETPCR.TEST must be set in to make the parity RAM accessible. Once HETPCR.TEST is set, the parity bits are accessible as described in [Table 19-4](#).

**Table 19-4. N2HET Parity Bit Mapping**

Address N2HET1	Address N2HET2	Bits	
		[31:1]	[0]
0xFF46_2000	0xFF44_2000	Reads 0, Writes have no effect	Instruction 0 Program Field Parity Bit
0xFF46_2004	0xFF44_2004	Reads 0, Writes have no effect	Instruction 0 Control Field Parity Bit
0xFF46_2008	0xFF44_2008	Reads 0, Writes have no effect	Instruction 0 Data Field Parity Bit

**Table 19-4. N2HET Parity Bit Mapping (continued)**

Address N2HET1	Address N2HET2	Bits	
		[31:1]	[0]
0xFF46_200C	0xFF44_200C	Reads 0, Writes have no effect	Read 0
0xFF46_2010	0xFF44_2010	Reads 0, Writes have no effect	Instruction 1 Program Field Parity Bit
....	....	...	...

Each 32-bit N2HET field has its own parity bit in the N2HET Parity RAM as shown in Table 19-4. There are no parity bits for the reserved fields, since there is no physical N2HET RAM for these fields.

**19.2.2.5 Initialization of Parity RAM**

After device power up, the N2HET RAM contents including the parity bits cannot be guaranteed. In order to avoid false parity failures due to the random state in which RAM powers up, the RAM has to be initialized.

Before initializing the N2HET RAM, enable the N2HET parity logic by writing to HETPCR. Then the N2HET Instruction RAM should be initialized. With parity enabled, the N2HET parity RAM will be initialized automatically by N2HET at the same time that the N2HET instruction RAM is initialized by the CPU. Note that loading the N2HET program with parity enabled is also effective.

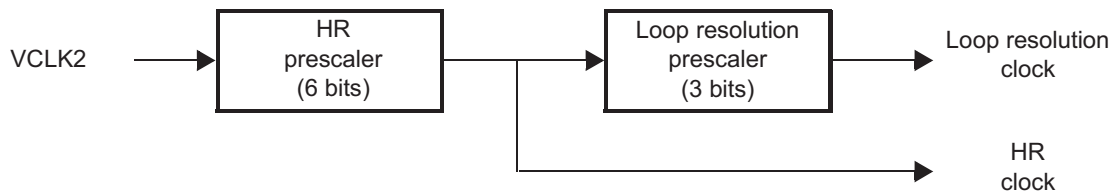
Another possibility to initialize the N2HET memory and its parity bits is, to use the system module to start the automatic initialization of all RAMs on the microcontroller. The RAMs will be initialized to '0'. Depending on the even/odd parity selection, the parity bit will be calculated accordingly.

**19.2.3 Time Base**

All N2HET timings are derived from VCLK2 (see Figure 19-7). Internally N2HET instructions execute at the VCLK2 rate; but the timer loop clock and the high-resolution hardware timer clock can be scaled down from VCLK2. Two prescalers are available to adjust the timer resolution clock for the program loop, and the high resolution (HR) clock for the HR I/O counters.

- Time Slots** The number of cycles available for instruction execution per loop. Time Slots is the number of VCLK2 cycles in a Loop Resolution Clock.
- High Resolution Clock** The high resolution clock is the smallest time increment with which a pin can change it's state or can be measured in the case of input signals. A 6-bit prescaler dividing VCLK2 by a user-defined HR prescale divide rate (hr) stored in the 6-bit HR prescale factor code (HETPFR). See Table 19-5.
- Loop Resolution Clock** The loop resolution clock defines the timebase for executing all instructions in a N2HET program. Since instructions can be conditionally executed, the longest path through the N2HET program must fit into one loop resolution clock period (LRP). A 3-bit prescaler dividing the HR clock by a user-defined loop-resolution prescale divide rate (lr) stored in the 3-bit loop-resolution prescale factor code (HETPFR). See Table 19-5.

**Figure 19-7. Prescaler Configuration**



The following abbreviations and relations are used in this document:

1. hr: high resolution prescale factor (1, 2, 3, 4,..., 63, 64)
2. lr: loop resolution prescale factor (1, 2, 4, 8, 16, 32, 64,128)
3. ts: Time slots (cycles) available for instruction execution per loop.  $ts = hr \times lr$
4. HRP = high resolution clock period  $HRP = hr \cdot T_{VCLK2}$  (ns)
5. LRP = loop resolution clock period  $LRP = lr \cdot HRP$  (ns)

The loop resolution period (LRP) must be selected to be larger than the number of Time slots (VCLK2 cycles) required to complete the worst-case execution path through the N2HET program. Otherwise a program overflow condition may occur (see [Section 19.2.1.4](#)). Because of the relationship of time slots to the hr and lr prescalers as described in item 3 above, increasing either hr or lr increases the number of time slots available for program execution. However, lr would typically be increased first, since increasing hr results in a decrease in timer resolution since it reduces the clock to the High Resolution IO structures.

The divide rates hr and lr can be defined in the HETPFR register. [Table 19-5](#) lists the bit field encodings for the prescale options.

**Table 19-5. Prescale Factor Register Encoding**

LRPFC - Loop Resolution		HRPFC - High Resolution	
HETPFR[10:8]	Prescale Factor lr	HETPFR[5:0]	Prescale Factor hr
000	/1	000000	/1
001	/2	000001	/2
010	/4	000010	/3
011	/8	000011	/4
100	/16	:::::	:
101	/32	111101	/62
110	/64	111110	/63
111	/128	111111	/64

### 19.2.3.1 Determining Loop Resolution

As an example, consider an application that requires high resolution of HRP = 62.5 ns, and loop resolution of LRP = 8 μs, and needs at least 250 time slots for the N2HET application program.

Assuming VCLK2 = 32 MHz, the following shows which divide-by rates and which value in the Prescale Factor Register (HETPFR) is required for the above requirements:

$$hr = 2 \rightarrow HRP = \frac{hr}{VCLK2} = \frac{2}{32MHz} = 62.5ns$$

$$lr = 128 \rightarrow lr \times HRP = 128 \times 62.5ns = 8 \mu s$$

$$ts = hr \times lr = 2 \times 128 = 256$$

$$hr = 2, lr = 128 \rightarrow HETPFR[31:0] = 0x00000701 \quad (29)$$

In the example above, if the loop resolution period needs to decrease from 8 μs to 4 μs, then only 128 time slots will be available for program execution. The program may need to be restructured as suggested in [Section 19.2.1.6](#), *Multi-Resolution Scheme*.

### 19.2.3.2 The 7-Bit HR Data Field

The instruction execution examples of ECMP ([Section 19.2.5.9](#)), MCMP ([Section 19.2.5.10](#)), PCNT ([Section 19.2.5.12](#)), PWCNT ([Section 19.2.5.11](#)), and WCAP ([Section 19.2.5.13](#)) show that the 7-bit HR data field can generate or measure high resolution delays (HR delay) relative to the start of an LRP within one N2HET loop LRP. The last section showed that:

$$LRP = lr \times HRP$$

There are lr high resolution clock periods (HRP) within the N2HET loop resolution clock period (LRP). If lr = 128 then the HR delay can range from 0 to 127 HRP clocks within LRP and all 7 bits of the HR data field are needed. Instead of being limited to measuring and triggering events based on the loop resolution clock period (LRP) the HR extension allows measurements and events to be described in terms fractions of an LRP (down to 1/128 of an LRP). The only limitation is that a maximum of one HR delay can be specified per pin during each loop resolution period.



Table 19-6 shows which bits of the HR data field are not used by the high resolution IO structures if  $l_r$  is less than 128. In this case the non-relevant bits (LSBs) of the HR data fields will be one of the following:

- Written as 0 for HR capture (e.g. for PCNT, WCAP)
- Or interpreted as 0 for HR compare (e.g. for ECMP, MCMP, PWCNT)

**Table 19-6. Interpretation of the 7-Bit HR Data Field**

Loop Resolution Prescale divide rate ( $l_r$ )	Bits of the HR data field <sup>(1)</sup>							HRP Cycles delay range
	D[6]	D[5]	D[4]	D[3]	D[2]	D[1]	D[0]	
1	X X X X X X X							0
2	1/2	X X X X X X						0 to 1
4	1/2	1/4	X X X X X				0 to 3	
8	1/2	1/4	1/8	X X X X			0 to 7	
16	1/2	1/4	1/8	1/16	X X X		0 to 15	
32	1/2	1/4	1/8	1/16	1/32	X X		0 to 31
64	1/2	1/4	1/8	1/16	1/32	1/64	X	0 to 63
128	1/2	1/4	1/8	1/16	1/32	1/64	1/128	0 to 127

<sup>(1)</sup> X = Non-relevant bit (treated as '0')

#### 19.2.3.2.1 Example:

Prescale Factor Register (HETPFR) = 0x0300

→  $l_r = 8$  →  $LRP = 8 \cdot HRP$

Assumption: HR data field = 0x50 = 1010000b

$l_r = 8$  → Bits D[3:0] are ignored → HR delay = 101b = 5 HRP

or by using the calculation with weight factors:

$$\begin{aligned}
 & \text{HR Delay} \\
 &= l_r \cdot (D[6] \cdot 1/2 + D[5] \cdot 1/4 + D[4] \cdot 1/8 + D[3] \cdot 1/16 + D[2] \cdot 1/32 + D[1] \cdot 1/64 + D[0] \cdot 1/128) \\
 &= 8 \cdot (1 \cdot 1/2 + 0 \cdot 1/4 + 1 \cdot 1/8 + 0 \cdot 1/16 + 0 \cdot 1/32 + 0 \cdot 1/64 + 0 \cdot 1/128) \\
 &= 5 \text{ HRP}
 \end{aligned}$$

### 19.2.4 Host Interface

The host interface controls all communications between timer-RAM and masters accessing the N2HET RAM. It includes following components:

#### 19.2.4.1 Host Accesses to N2HET RAM

The host interface supports the following types of accesses to N2HET RAM:

- Read accesses of 8, 16, or 32 bits
- Read accesses of 64-bits that follow the shadow register sequence described in [Section 19.2.4.2](#).
- Write accesses of 32 bits

Writes of 8 or 16 bits to N2HET RAM by an external host are not supported.

#### 19.2.4.2 64-bit Read Access

The consecutive read of a control field CF(n) and a data field DF(n) of the same instruction (n) performed by the same master (e.g. CPU, DMA or any other master) is always done as a simultaneous 64-bit read access. This means that at the same time CF(n) is read, DF(n) is loaded in a shadow register. So the second access will read DF(n) from the shadow register instead of the N2HET RAM.



In general a 64-bit read access of one master could be interrupted by a 64-bit read access of another master. A total of three shadow registers are available. Therefore up to three masters can perform 64-bit reads in an interleaved manner (e.g. Master1 CF, Master2 CF, Master3 CF, Master1 DF, Master2 DF, Master3 DF).

If all three shadow registers are activated and a 4th master performs a CF or DF read it will result in an address error and the RAM access will not happen. Other access types by a fourth master (reads from the PF field or writes to any of the fields) will occur because these access types do not require an available shadow register resource to complete.

#### 19.2.4.3 Automatic Read Clear Feature

The N2HET provides a feature allowing to automatically clear the data field immediately after the data field is read by the external host CPU (or DMA). This feature is implemented via the *control bit*, which is located in the control field (bit C26). This is a static bit that can be used by any instruction, and specified in the N2HET program by adding the option (control = ON) to the N2HET instruction. The automatic read clear feature works for both 32 and 64 bit reads that follow the sequence described in [Section 19.2.4.2](#).

When the host CPU reads the data field of that instruction, the current data value is returned to the host CPU but the field is cleared automatically as a side effect of the read. In case the master reads data from an instruction currently executing, any new capture result is stored and this takes priority over the automatic read clear feature, so that the new capture result is not lost.

As an example of where the automatic read clear feature is useful, consider the PCNT instruction. If this instruction is configured for automatic read clear, then when the host CPU reads the PCNT data field it will be cleared automatically. The host CPU can then poll the PCNT data field again, and as long as the field returns a value of zero the host CPU program knows a new capture event has not occurred. If the data field were not cleared, it would be impossible for the host CPU to determine whether the data field holds data from the previous capture event, or if it happens to be data from a new capture event with the same value.

#### 19.2.4.4 Emulation Mode

Emulation mode, used by the software debugger, is specified in the global configuration register. When the host CPU debugger hits a breakpoint, the CPU sends a suspend signal to the modules. Two modes of operation are provided: suspend and ignore suspend.

- Suspend

When a suspend is issued, the timer operation stops at the end of the current timer instruction. However, the CPU accesses to the timer RAM or control registers are freely executed.

- Ignore suspend

The timer RAM ignores the suspend signal and operates real time as normal.

#### 19.2.4.5 Power-Down

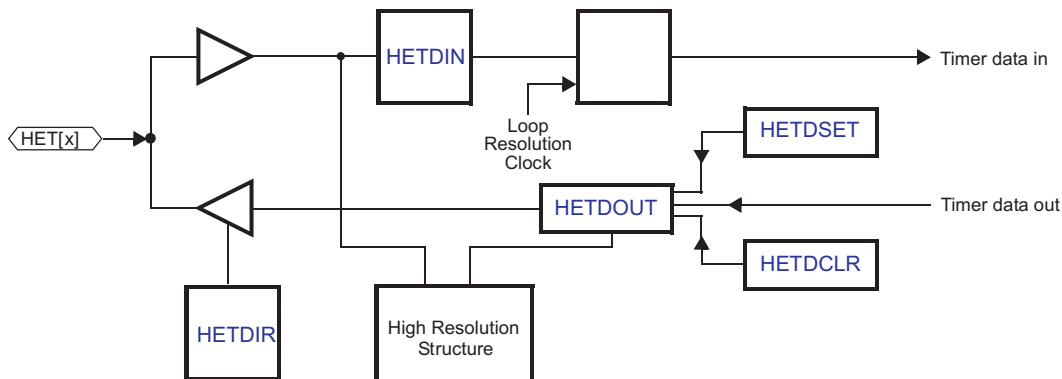
After setting the turn-off bit in the Global Configuration Register (HETGCR), it is required to delay until the end of the timer program loop before putting the N2HET in power-down mode. This can be done by waiting until the N2HET Current Address (HETADDR) becomes zero, before disabling the N2HET clock source in the device's Global Clock Module (GCM).

### 19.2.5 I/O Control

The N2HET has up to 32 pins. Refer to device specific data sheets for information concerning the number of N2HETIO available. All of the N2HET pins available are programmable as either inputs or outputs.

These 32 I/Os have an identical structure connected to pins HET[31] to HET[0]. See [Figure 19-8](#) for an illustration of the I/O control. In addition all 32 I/Os have a special HR structure based on the HR clock. This structure allows any N2HET instruction to use any of these I/Os with an accuracy of either loop resolution or high resolution accuracy.

Figure 19-8. I/O Control



Pins N2HET [31] to N2HET [0] can be used by the CPU as general purpose inputs or outputs using the N2HET Data Input Register (HETDIN) for reading and N2HET Data Output Register (HETDOUT), N2HET Data Set Register (HETDSET) or N2HET Data Clear Register (HETDCLR) for writing, depending on the type of action to perform. The N2HET pins used as general purpose inputs are sampled on each VCLK2 period.

### 19.2.5.1 Using General Purpose I/O Data Set and Clear Registers

The N2HET Data Clear Register (HETDCLR) and N2HET Data Set Register (HETDSET) can be used to minimize the number of accesses to the peripheral to modify the output register and output pins. When the application needs to set or to reset some N2HET pins without changing the value of the others pins, the first possibility is to read N2HET Data Output Register (HETDOUT), modify the content (AND, OR, etc.), and write the result into N2HET Data Output Register (HETDOUT). However, this read-modify-write sequence could be interrupted by a different function modifying the same register which will result in a data coherency problem.

Using the N2HET Data Set Register (HETDSET) or N2HET Data Clear Register (HETDCLR), the application program must write the mask value (same mask value for the first option) to the register to set or reset the desired pins. Any bits written as 0 to HETDSET and HETDCLR are left unchanged, which avoids the possible coherency problem of the read-modify-write approach.

```
Coding Example (C program): Set pins using the 2 methods.
unsigned int MASK; /* Variable that content the bit mask */
volatile unsigned int *HETDOUT,*HETDSET; /* Pointer to HET registers */
...
*HETDOUT = *HETDOUT | MASK; /* Read-modify-write of HETDOUT */
*HETDSET = MASK; /* Set the pin without reading HETDOUT */
```

### 19.2.5.2 Loop Resolution Structure

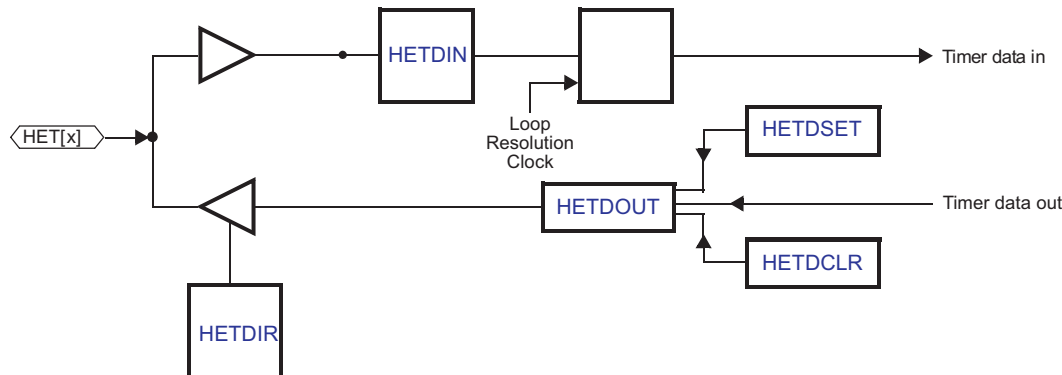
The N2HET uses the pins N2HET [31:0] as input and/or output by the way of the instruction set. Actually, each pin could monitor the N2HET program or could be monitored by the N2HET program. By using the I/O register of the N2HET, the CPU is able to interact with the N2HET program flow.

When an action (set or reset) is taken on a pin by the N2HET program, the N2HET will modify the pin at the rising edge of the next resolution clock.

When an event occurs on a N2HET 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 19-9](#).

**Figure 19-9. N2HET Loop Resolution Structure for Each Bit**



The example in [Figure 19-10](#) shows a simple PWM generation with loop resolution accuracy. The corresponding program can be found below.

HETPFR[31:0] register = 0x201 --> lr=4 and hr=2 --> ts = 8

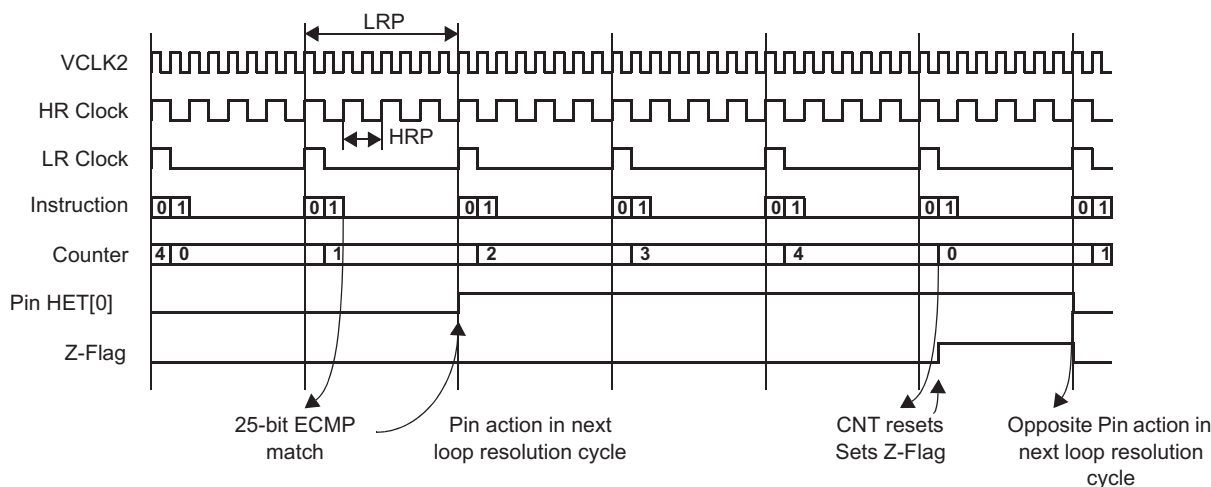
**N2HET Program:**

```
L00 CNT { next= L01, reg=A, irq=OFF, max = 4 }
L01 ECMP { next= L00, cond_addr= L00, hr_lr=LOW, en_pin_action=ON, pin=0,
          action=PULSEHI, reg=A, irq=OFF, data= 1, hr_data = 0x0 }
```

; 25 bit compare value is 1 and the 7-bit HR compare value is 0

The CNT and ECMP instructions are executed once each loop resolution cycle. When the CNT instruction is executed, the specified register (A) and the CNT instruction data field are both incremented by one. Next the ECMP is executed and the data field of the ECMP is compared with the specified register (A). If both values match, then the pin action (PULSEHI in this case) will be performed in the next loop resolution cycle. The CNT continues incrementing each loop resolution cycle. When the data field overflows (max + 1), then the Z-flag is set by the CNT instruction. In the next loop resolution cycle, the Z-flag is evaluated and the opposite pin action is performed if it is set. The Z-flag will only be active for one loop resolution cycle.

**Figure 19-10. Loop Resolution Instruction Execution Example**



### 19.2.5.3 High Resolution Structure

All 32 I/Os provide the HR structure based on the HR clock. The HR clock frequency is programmed through the Prescale Factor Register (HETPFR). In addition to the standard I/O structure, all pins have HR hardware so that these pins can be used as HR input captures (using the HR instructions PCNT or WCAP) or HR output compares (using the HR instructions ECMP, MCMP or PWCNT).

All five HR instructions have a dedicated hr\_lr bit (high resolution/low resolution; program field bit 8) allowing operation either in HR mode or in standard resolution mode by ignoring the HR field. By default, the hr\_lr bit value is 0 which implies HR operation mode. However, setting this bit to one allows the use of several HR instructions on a single HR pin. Only one instruction is allowed to operate in HR mode (i.e., bit set to 0), but the other instructions can be used in standard resolution mode (i.e., bit set to 1).

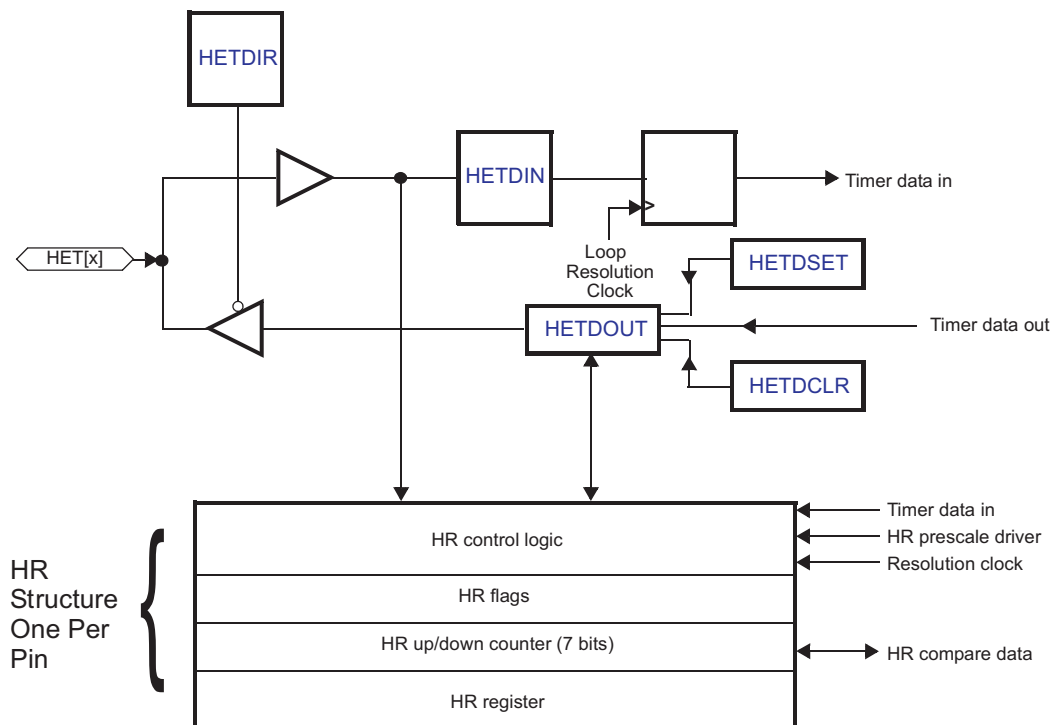
### 19.2.5.4 HR Block Diagram

Each time an HR instruction is executed on a given pin, the HR structure for that pin is programmed (which HR function to perform and on which edges it should take an action) with the information given by the instruction. The HR structure for each pin decodes the pin select field of the instruction and programs its HR structure if it matches.

**NOTE:** For each N2HET pin, only one instruction specifying a high resolution operation (hr\_lr=HIGH) is allowed to execute per loop resolution period. This includes any instructions where (hr\_lr=HIGH) but (en\_pin\_action=OFF).

**NOTE:** The first high resolution instruction that executes and specifies a particular pin locks out subsequent high resolution instructions from operating on the same pin until the end of the current loop resolution period.

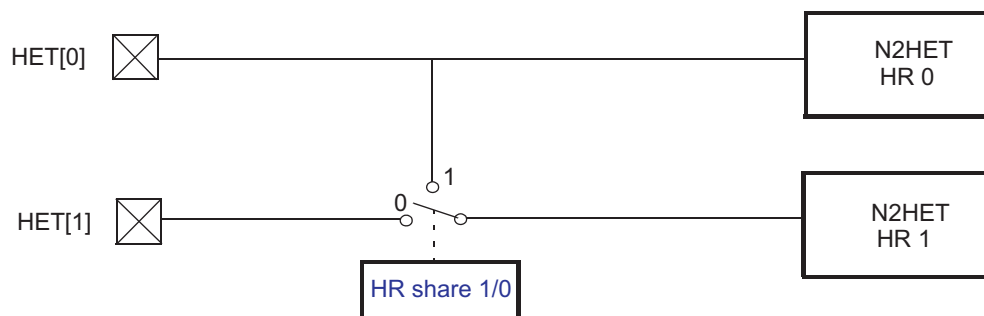
Figure 19-11. HR I/O Architecture



### 19.2.5.5 HR Structures Sharing (Input)

The HR Share Control Register (HETHRSH) allows two HR structures to share the same pin for **input capture only**. If these bits are set, the HR structures N and N+1 are connected to pin N. In this structure, pin N+1 remains available for general purpose input/output. See [Figure 19-12](#).

**Figure 19-12. Example of HR Structure Sharing for N2HET Pins 0/1**



The following program gives an example how the HR share feature (HET[0] HR structure and HET[1] HR structure shared) can be used for the PCNT instruction:

```
L00 PCNT { next=L01, type=rise2fall, pin=0 }
L01 PCNT { next=L00, type=fall2rise, pin=1 }
```

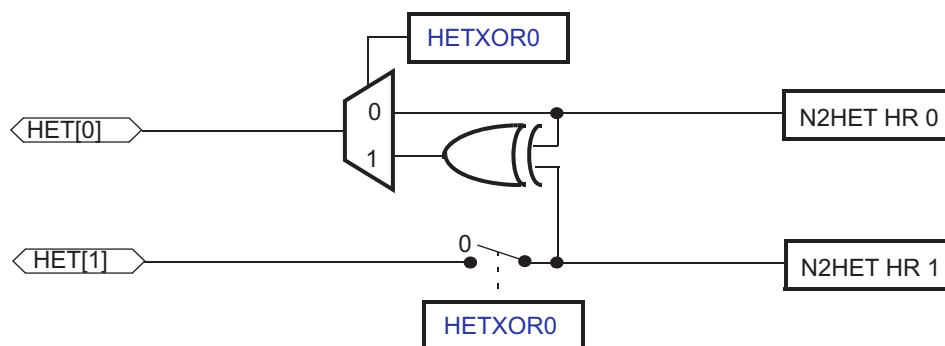
The HET[1] HR structure is also connected to the HET[0] pin. The L00\_PCNT data field is able to capture a high pulse and the L01\_PCNT captures a low pulse on the **same** pin (N2HET [0] pin).

### 19.2.5.6 AND / XOR-shared HR Structure (Output)

Usually the N2HET design allows only one HR structure to generate HR edges on a pin configured as output pin. The HETXOR register allows a logical XOR of the output signals of two consecutive HR structures N (even) and N+1 (odd). See [Figure 19-13](#). In this way, it is possible to generate pulses smaller than the loop resolution clock since both edges can be generated by two independent HR structures. This is especially required for symmetrical PWM. See [Figure 19-14](#).

The hardware provides a XOR gate that is connected to the outputs of the HR structure of two consecutive pins. In this structure, pin N+1 remains available for general purpose input/output.

**Figure 19-13. XOR-shared HR I/O**



The following N2HET program gives an example for **one** channel of the symmetrical PWM. The generated timing is given in [Figure 19-14](#).

```
MAXC .equ 22
A_ .equ 0 ; HR structure HR0
B_ .equ 1 ; HR structure HR1

CN CNT { next=EA, reg=A, max=MAXC }
```

```
EA ECMP { next=EB, cond_addr=MA, hr_lr=HIGH, en_pin_action=ON, pin=A_,
          action=PULSELO, reg=A, data=17, hr_data=115 }

MA MOV32 { next=EB, remote=EA, type=IMTOREG&REM, reg=NONE, data=17, hr_data=19 }

EB ECMP { next=CN, cond_addr=MB, hr_lr=HIGH, en_pin_action=ON, pin=B_,
          action=PULSELO, reg=A, data=5, hr_data=13 }

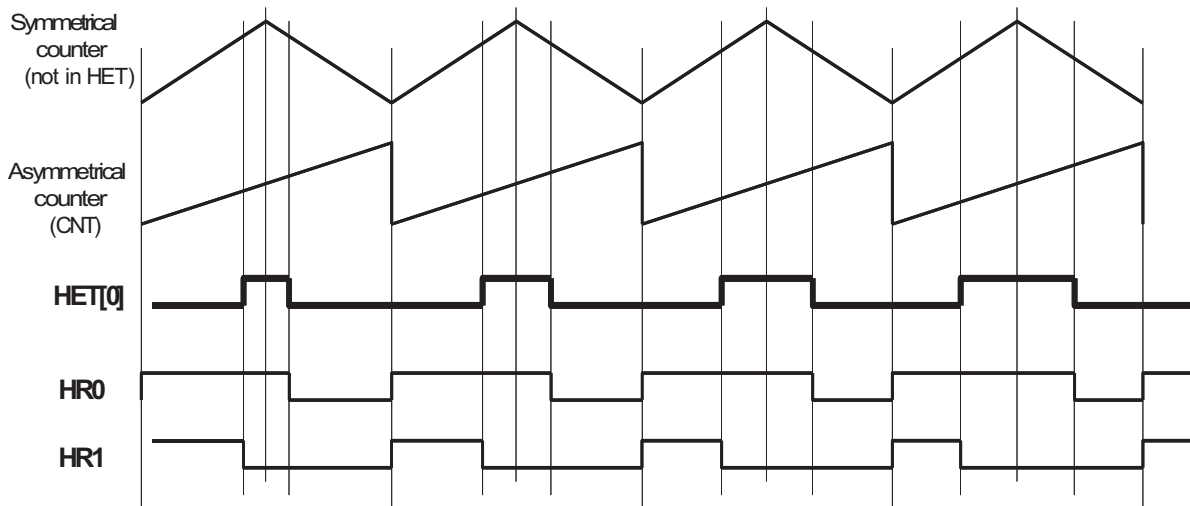
MB MOV32 { next=CN, remote=EB, type=IMTOREG&REM, reg=NONE, data=5, hr_data=13 }
```

N2HET Settings and output signal calculation for this example program:

- Pin HET[0] and HET[1] are XOR-shared.
- HETPFR[31:0] register = 0x700: lr=128, hr=1, time slots ts = 128
- PWM period (determined by CNT\_max field) = (22+1) · LRP = 2944 HRP
- Length of high pulse of (HET[0] XOR HET[1]) =  
 $LH = (17 \cdot LRP + 115 \cdot HRP) - (5 \cdot LRP + 13 \cdot HRP)$   
 With lr=128 there is  $LRP = 128 \cdot HRP$ , so  
 $LH = (2291 - 653) \cdot HRP = 1638 \text{ HRP}$
- Duty cycle = DC =  $LH / \text{PWM\_period} = 1638 \text{ HRP} / (2944 \cdot \text{HRP}) = 55.6 \%$

Figure 19-14 graphically shows the implementation of the XOR-shared feature. The first 2 waveforms (symmetrical counter and CNT) show a symmetric counter and asymmetric counter. The symmetric counter is shown only to highlight the axis of symmetry and is not implemented in the N2HET. The asymmetric counter, which is implemented with a CNT instruction, needs to be set to the period of the symmetric counter. The next two waveforms (HR [0] and HR [1]) show the output of the HR structures, which are the inputs for the XOR gate to create the PWM output on pin HET[0]. Notice that the pulses of signal HET[0] are centered about the axis of symmetry.

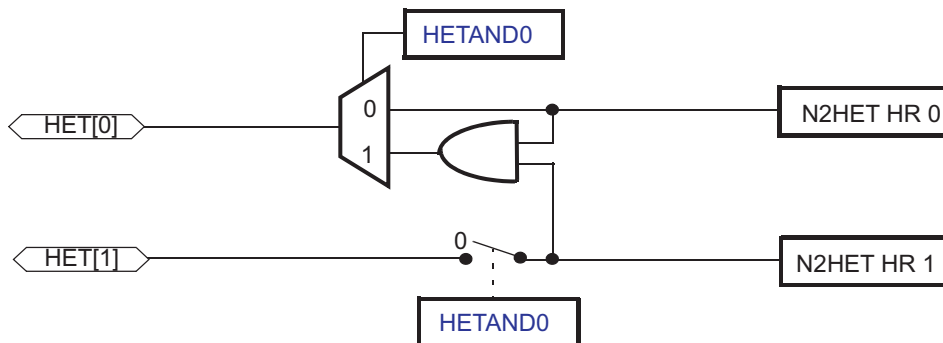
Figure 19-14. Symmetrical PWM with XOR-sharing Output



As an alternative, HR structures may be shared using a logical AND function to combine the effects of the pin structures. The HETAND allows sharing two consecutive HR structures N (even) and N+1 (odd). See Figure 19-15. In this structure, pin N+1 remains available for general purpose input/output.

**NOTE:** Setting both the HETAND bit and HETXOR bits at the same time for a given pair of N2HET pins is not supported, must be avoided by the application program.

Figure 19-15. AND-shared HR I/O



### 19.2.5.7 Loop Back Mode

The loop back feature can be used by the application to monitor an N2HET output signal. For example, if a PWM is generated by HR structure 0, then a PCNT instruction assigned to HR structure 1 can measure back the pulse length or periods of the PWM output signal.

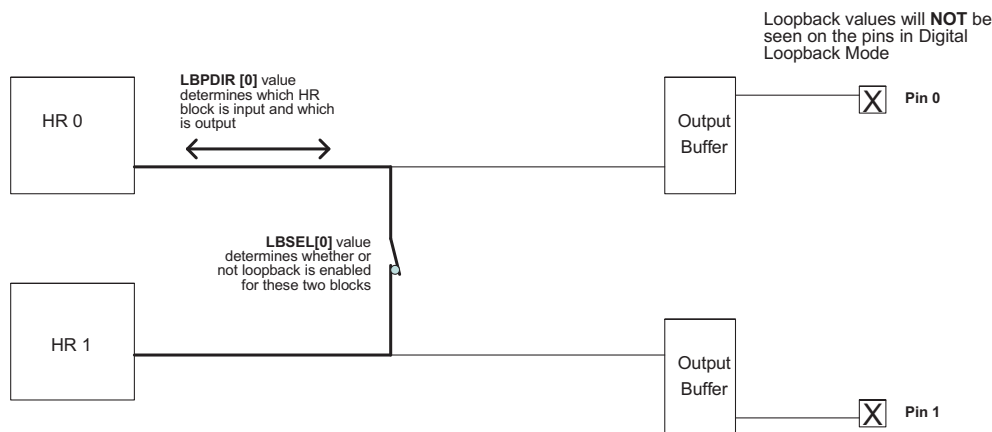
Loopback mode is activated between two high resolution structures by setting LBPSEL[x] to '1' in the HETLBPSEL Register for the corresponding structure pair. The **direction** of the loopback between the two structures in the structure pair is determined by the value of LBPDIR[x] in the HETLBPDIR Register.

For example, if bit LBPSEL[0] is set to '1', then HR structures 0 and 1 will be internally connected in loop back mode. If bit LBPDIR[0] is set to '0' then structure 0 will be the input and structure 1 will be the output.

#### Digital Loopback

Digital loopback mode is enabled by setting LBPTYPE[x] to '0' in the HETLBPSEL Register for the corresponding structure pairs. In digital loopback mode, the structure pairs are connected directly and the output buffers are bypassed. Therefore, the loopback values will NOT be seen on the corresponding pins. Figure 19-16 below shows an example of digital loopback between structures HR0 and HR1. LBSEL[0] has been set to '1' to enable loopback between the two structures. LBTYPE[0] has been set to '0' to select digital mode for the loopback pair. The LBPDIR[0] value will determine the direction of the loopback by selecting which of the HR blocks is output, and which is input. The bold lines show the digital loopback path.

Figure 19-16. HR0 to HR1 Digital Loopback Logic: LBTYPE[0] = '0'

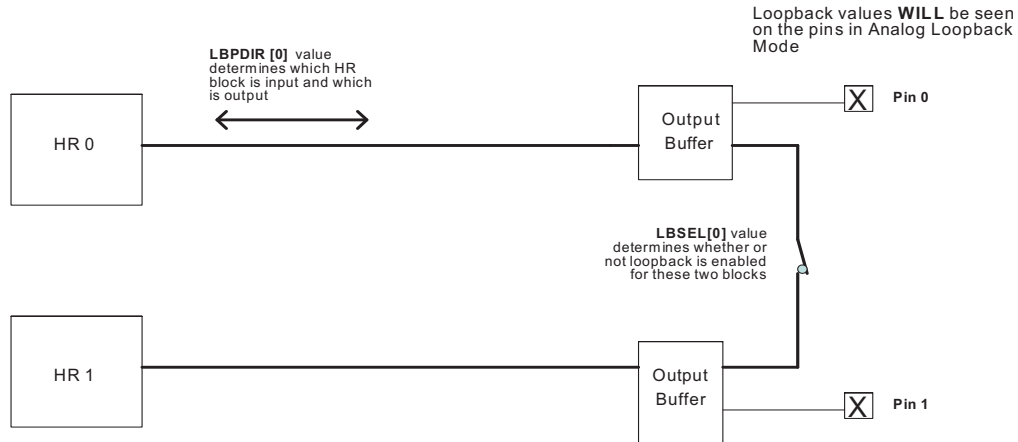


#### Analog Loopback

Analog loopback mode is enabled by setting LBPTYPE[x] to '1' in the HETLBPSEL Register for the corresponding structure pairs. In analog loopback mode, the structure pairs are connected outside of the

output buffers. Therefore, the loopback values WILL be seen on the corresponding pins. Figure 19-17 below shows an example of analog loopback between structures HR0 and HR1. LBSEL[0] has been set to '1' to enable loopback between the two structures. LBTYPE[0] has been set to '1' to select analog mode for the loopback pair. The LPBDIR[0] value will determine the direction of the loopback by selecting which of the HR blocks is output, and which is input. The bold lines show the analog loopback path.

Figure 19-17. HR0 to HR1 Analog Loop Back Logic: LBTYPE[0] = '1'



Note:

- The loop back direction can be selected independent of the HETDIR register setting.
- The pin which is not driven by the N2HET output pin actions can still be used as normal GIO pin.

### 19.2.5.8 Edge Detection Input Timing

There are several timing requirements for input signals in order to be captured correctly by N2HET. Figure 19-18 illustrates these requirements, with min and max values described in Table 19-7 (Loop Resolution) and Table 19-8 (High Resolution).

Figure 19-18. N2HET Input Edge Detection

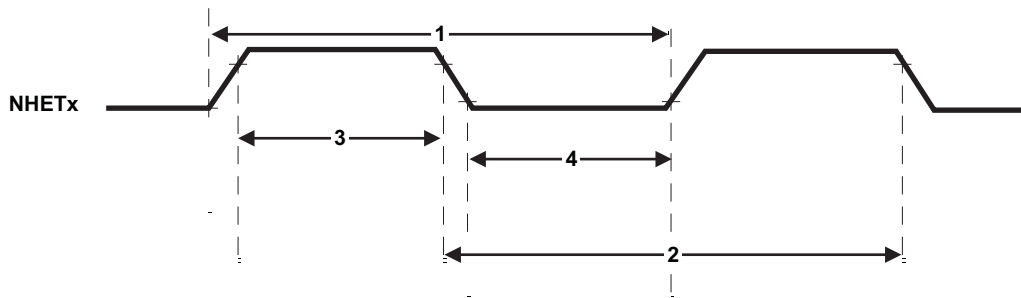


Table 19-7. Edge Detection Input Timing for Loop Resolution Instructions

Parameter #	Description	min	max
1	Input Signal Period, rising edge to rising edge	> 2 (hr) (lr) $t_{c(VCLK2)}$	< 2 <sup>25</sup> (hr) (lr) $t_{c(VCLK2)}$
2	Input Signal Period, falling edge to falling edge		
3	Input Signal, high phase	> (hr) (lr) $t_{c(VCLK2)}$	
4	Input Signal, high phase		



**Table 19-8. Edge Detection Input Timing for High Resolution Instructions**

Parameter #	Description	min	max
1	Input Signal Period, rising edge to rising edge	> (hr) (lr) $t_{c(VCLK2)}$	< $2^{25}$ (hr) (lr) $t_{c(VCLK2)}$
2	Input Signal Period, falling edge to falling edge		
3	Input Signal, high phase	> 2 (hr) $t_{c(VCLK2)}$	
4	Input Signal, high phase		

These are the N2HET architectural limitations. Actual limitations will be slightly different due to on chip routing and IO buffer delays, usually by several nanoseconds. Be sure to consult the device datasheet for actual timings that apply to that device. Also, certain devices place additional restrictions on which pins support the high resolution timings of [Table 19-8](#), if present these additional limitations will also be called out in the device datasheet.

Note that the max limit in [Table 19-7](#) and [Table 19-8](#) is based on the counter range of a single N2HET instruction. The max value could be extended by employing an additional N2HET instruction to keep track of counter overflows of the input counter / capture instruction.

### 19.2.5.9 PWM Generation Example 1 (in HR Mode)

The following example shows how an ECMP instruction works in high resolution mode. The example assumes a VCLK2 of 32 MHz and the following values for the prescale divide rates (hr and lr), number of time slots (ts), high and loop resolution period (HRP and LRP):

$$hr = 2, lr = 4, ts = hr \cdot lr = 8$$

$$HRP = hr / VCLK2 = 2 / 32 \text{ MHz} = 62.5 \text{ ns}$$

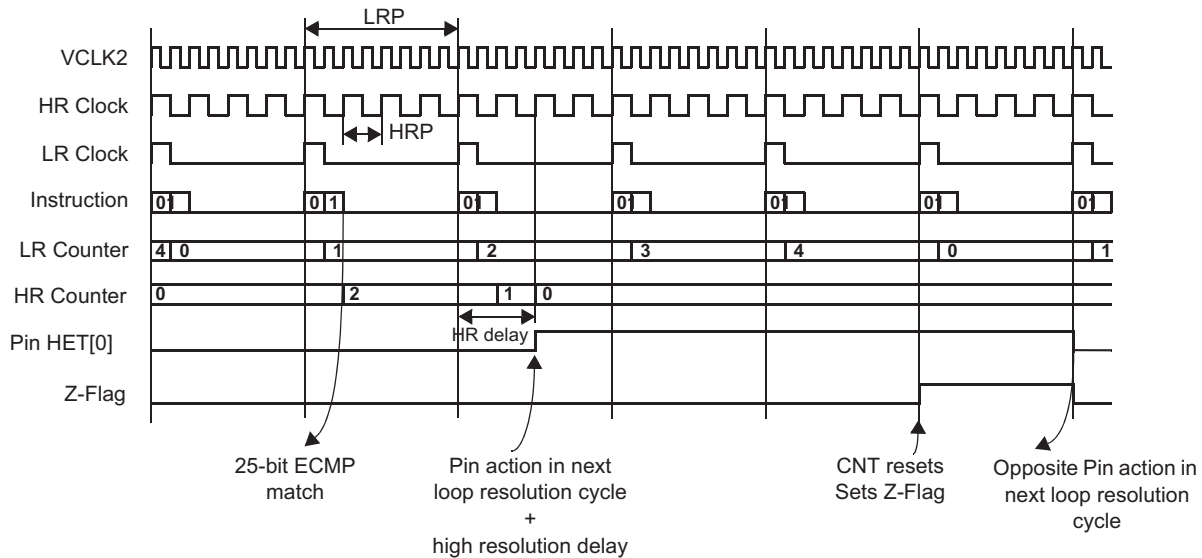
$$LRP = (hr \cdot lr) / VCLK2 = 8 / 32 \text{ MHz} = 250 \text{ ns}$$

With  $ts=8$  there are eight time slots available for the program execution, which in this case will consist of one CNT and one ECMP instruction as shown below. The data field of the ECMP instruction is the 32-bit compare value, whereby the lower 7 bits represent the high resolution compare field.

When the 25-bit (loop resolution) compare matches, the HR compare value will be loaded from the 7 lower bits of the instruction data field to the HR counter. At the next resolution clock, the HR counter will count down at the HR clock frequency and perform the pin action when it reaches zero.

In the example illustrated by [Figure 19-19](#) the 25-bit compare value is one and the 7-bit HR compare value is two. According to [Section 19.2.3.2](#), depending on the loop resolution divide rate (lr), only certain bits of the 7-bit HR compare value are valid. In this example only the upper 2 bits (D[6:5]) are taken into account. The example program below has a setting of  $hr\_data = 100000b$ . Shifting this value right by 5 bits, results in 10b which equals the two HR clock cycles delay mentioned above.

Figure 19-19. ECMP Execution Timings



HETPFR[31:0] register = 0x201 --> lr=4 and hr=2 --> ts = 8

**N2HET Program:**

```
L00  CNT  { next= L01, reg=A, irq=OFF, max = 4 }
L01  ECMP { next= L00, cond_addr= L00, hr_lr=HIGH, en_pin_action=ON, pin=0,
         action=PULSEHI, reg=A, irq=OFF, data= 1, hr_data = 0x40 }
```

; 25 bit compare value is 1 and the 7-bit HR compare value is 2  
 ; (Because of lr=4 the D[4:0] of the 7-bit HR field are ignored )

**NOTE: ECMP Opposite Actions**

ECMP opposite pin actions are always synchronized to the loop resolution clock.

Changing the duty cycle of a PWM generated by an ECMP instruction, can lead to a missing pulse if the data field of the instruction is updated directly. This can happen when it is changed from a high value to a lower value while the CNT instruction has already passed the new updated lower value. To avoid this a synchronous duty cycle update can be performed with the use of an additional instruction (MOV32). This instruction is only executed when the compare of the ECMP matches. For this the cond\_addr of the ECMP needs to point to the MOV32. On execution of the MOV32, it moves its data field into the data field of the ECMP. The update of the duty cycle has to be made to the MOV32 data field instead of the ECMP data field.

**19.2.5.10 PWM Generation Example 2 (in HR Mode)**

The MCMP instruction can also be used in HR mode. In this case operation is exactly the same as for the ECMP instruction except that the 25-bit low resolution is now the result of a magnitude compare (greater or equal) rather than an equality compare. When the 25-bit (loop resolution) magnitude compare matches, the HR compare value will be loaded from the 7 lower bits of the instruction data field to the HR counter. At the next resolution clock, the HR counter will count down at the HR clock frequency and perform the pin action when it reaches zero.

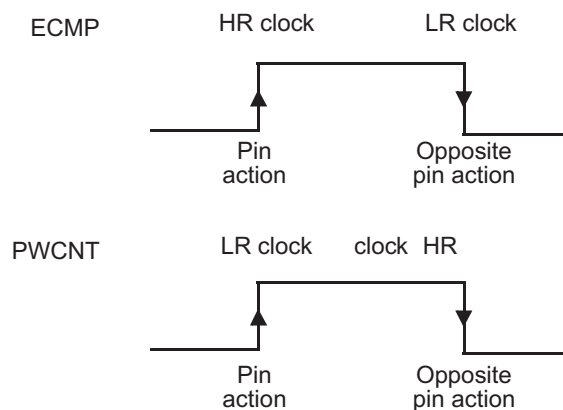
The MCMP instruction avoids the missing pulse problem of the ECMP instruction (see previous example), however the duty cycle of the signal might not be exact for one PWM period. The benefit of the MCMP is that it avoids adding another instruction to do the duty cycle update synchronously.

### 19.2.5.11 Pulse Generation Example (in HR Mode)

The PWCNT instruction may also be used in HR mode to generate pulse outputs with HR width. It generates a single pulse when the data field of the instruction is non-zero. It remains at the opposite pin action when the data field is zero.

The PWCNT instruction operates conversely to the ECMP instruction. See [Figure 19-20](#). For PWCNT, the opposite pin action is synchronous with the HR clock and for ECMP the pin action is synchronous with the HR clock. The PWCNT pin action is synchronous with the loop resolution clock.

**Figure 19-20. High/Low Resolution Modes for ECMP and PWCNT**



### 19.2.5.12 Pulse Measurement Example (in HR Mode)

The PCNT instruction captures HR measurement of the high/low pulse time or periods of the input. As shown in [Figure 19-21](#), at marker (1) the input goes HIGH and the HR counter immediately begins to count. The counter increments and rolls over until the falling edge on the input pin, where it captures the counter value into the HR capture register (marker (2)). The PCNT instruction begins counting when the synchronized input signal goes HIGH and captures both the 25-bit data field and the HR capture register into RAM when the synchronized input falls (marker (3)).

---

**NOTE:** The HR capture value written into RAM is shifted appropriately depending on the loop resolution prescale divide rate ( $I_r$ ). (See also [Section 19.2.3.2](#), "The 7-bit HR data field")

---

[Figure 19-21](#) shows what happens when the capture edge arrives *after* the HR counter overflows. This causes the incremented value to be captured by the PCNT instruction.

**Figure 19-21. PCNT Instruction Timing (With Capture Edge After HR Counter Overflow)**

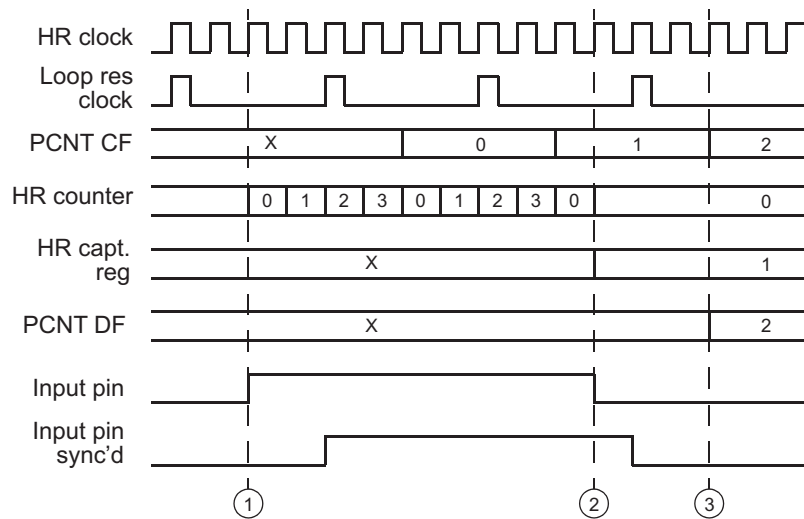
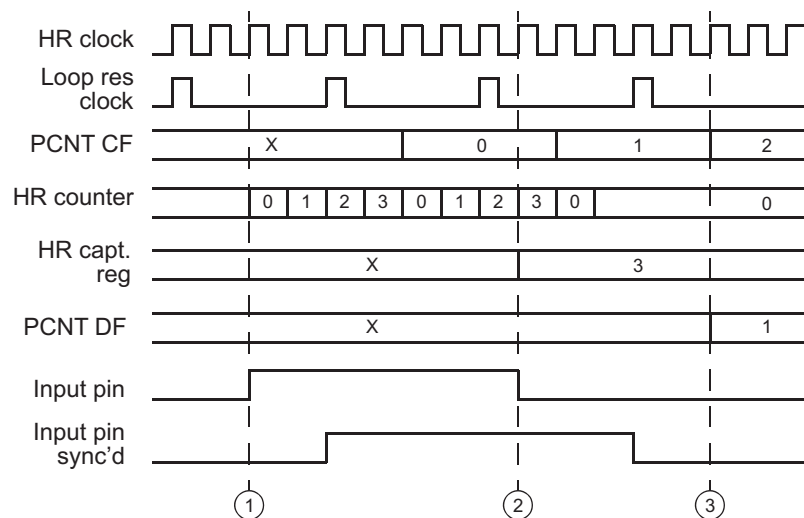


Figure 19-22 shows what happens when the capture edge arrives *before* the HR counter overflows. This causes the non-incremented value to be captured by the PCNT instruction.

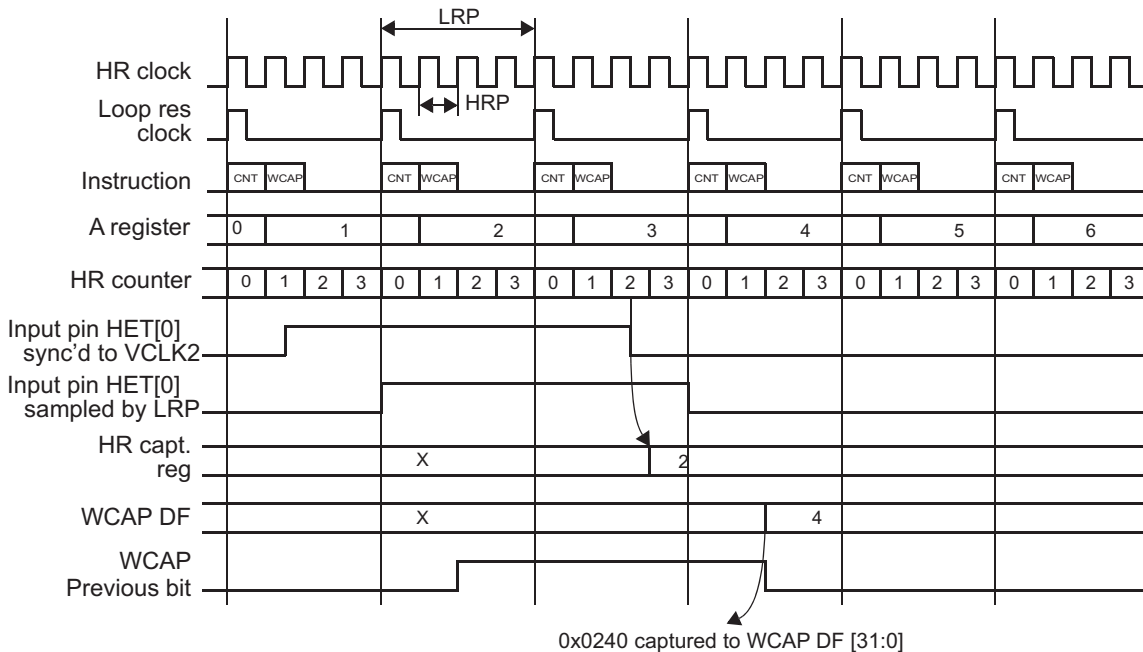
**Figure 19-22. PCNT Instruction Timing (With Capture Edge Before HR Counter Overflow)**



### 19.2.5.13 WCAP Execution Example (in HR Mode)

The HR capability is enabled for WCAP, if its hr\_lr bit is zero. In this case the HR counter is always enabled and is synchronized with the resolution loop. When the specified edge is detected, the current value of the HR counter is captured in the HR capture register and written into the RAM after the next WCAP execution. The WCAP instruction effectively time stamps the free running timer saved in a register (for example, register A shown in Figure 19-23).

**Figure 19-23. WCAP Instruction Timing**



HETPFR\_register = 0x0200 --> lr = 4, hr = 1, ts = 4

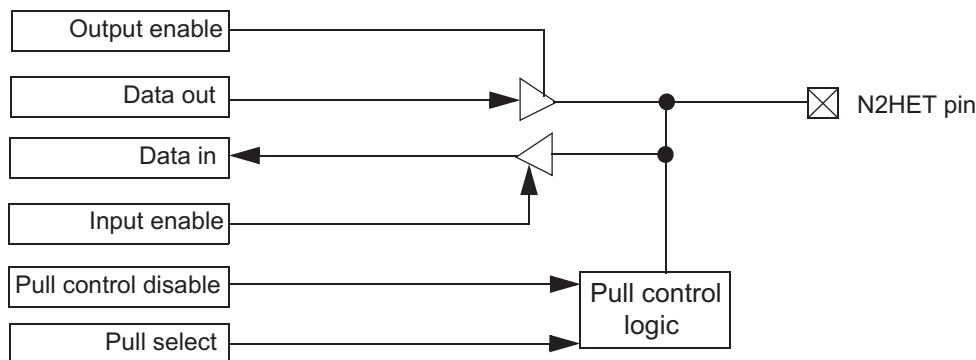
**N2HET Program:**

```
L00 CNT {reg=A, max=01ffffffh}
L01 WCAP {next=L00, cond_addr=L00, hr_lr=high, reg=A, event= FALL, pin=0,
data=0}
```

In the example, the WCAP is configured to capture the counter when a **falling** edge occurs. The WCAP data field (WCAP\_DF) is updated in the loop succeeding the loop in which the edge occurred. The WCAP instruction evaluates an edge by comparing its Previous bit with the sync'd input signal. In [Figure 19-23](#), the current value of the counter (4) is captured to WCAP\_DF[31:7] and the value of the HR capture register (2) is transferred to the valid bits (according the lr prescaler) of WCAP\_DF[6:0]. Therefore, in the example 0x0240 is captured in WCAP\_DF[31:0].

**19.2.5.14 I/O Pull Control Feature**

**Figure 19-24. I/O Block Diagram Including Pull Control Logic**



The following apply if the device is under reset:

- Pull control: The reset pull control on the pins is enabled and a pulldown is configured.
- Input buffer: The input buffer is enabled.

- Output buffer: The output buffer is disabled.

The following apply if the device is out of reset:

- Pull control: The pull control is enabled by clearing the corresponding bit in the N2HET Pull Disable Register (HETPULDIS). In this case, if the corresponding bit in the N2HET Pull Select Register (HETPSL) is set, the pin will have a pull-up; if the bit in the N2HET Pull Select Register (HETPSL) is cleared, the pin will have a pull-down. If the bit in the N2HET Pull Disable Register (HETPULDIS) is set, there is no pull-up or pull-down on the pin.
- Input buffer: The input buffer is disabled only if the pin direction is set to input AND the pull control is disabled AND pull down is selected as the pull bias. In all other cases, the input buffer is enabled.

---

**NOTE:** The pull-disable logic depends on the pin direction. If the pin is configured as output, then the pulls are disabled automatically. If the pin is configured as input, the pulls are enabled or disabled depending on the pull disable register bit.

---

- Output buffer: A pin can be driven as an output pin if the corresponding bit in the N2HET Direction Register (HETDIR) is set AND the open-drain feature (N2HET Open Drain Register (HETPDR)) is not enabled. See [Section 19.2.5.15, Open-Drain Feature](#) for more details.

The behavior of the input buffer, output buffer, and the pull control is summarized in [Table 19-9](#). When an input buffer is disabled, it appears as a logic low to on-chip logic.

**Table 19-9. Input Buffer, Output Buffer and Pull Control Behavior <sup>(1)</sup>**

Device under Reset?	Pin Direction (DIR)	Pull Disable (PULDIS)	Pull Select (PULSEL)	Pull Control	Output Buffer	Input Buffer
Yes	X	X	X	Device- and module-specific	Disabled	Disabled
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

<sup>(1)</sup> X = Don't care

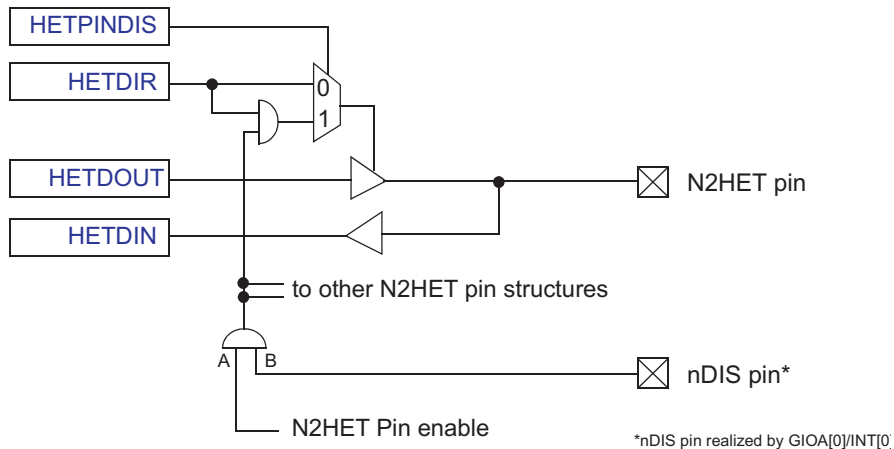
### 19.2.5.15 Open-Drain Feature

The following apply if the open-drain feature is enabled on a pin, i.e. the corresponding bit in the N2HET Open Drain Register (HETPDR) is set

- Output buffer is enabled if a low signal is being driven internally to the pin.
- The output buffer is disabled if a high signal is being driven internally to the pin.

### 19.2.5.16 N2HET Pin Disable Feature

This feature is provided for the safe operation of systems such as power converters and motor drives. It can be used to inform the monitoring software of motor drive abnormalities such as over-voltage, over-current, and excessive temperature rise.

**Figure 19-25. N2HET Pin Disable Feature Diagram**

The following truth-table shows the conditions for the output buffer to be enabled/disabled.

**Table 19-10. N2HET Pin Disable Feature**

HETPINDIS.x	nDIS Pin (Input)	HET PIN ENA (HETGCR.24)	HETDIR.x	Output Buffer
0	X	X	0	Disabled
0	X	X	1	Enabled
1	0	X	0	Disabled
1	0	X	1	Disabled
1	1	X	0	Disabled
1	1	0	1	Disabled
1	1	1	1	Enabled

An interrupt capable device I/O pin (GIOA[0]/INT[0]) shares the same pin as the N2HET nDIS signal. This allows the event on the N2HET nDIS input to also generate an interrupt to the CPU through the GIO module. An active low level on nDIS is intended to signal an abnormal situation as described above. All N2HET pins, which are selected with the N2HET Pin Disable Register (HETPINDIS), will be put in the high-impedance state by hardware immediately after the nDIS signal is pulled low. At this time a CPU interrupt is issued, if it is enabled in the GIO pin logic.

The bit HET PIN ENA is automatically cleared in the failure condition and this state remains as long as the software explicitly sets the bit again. The steps to do this are:

- Software detects, by reading the HETDIN register of the GIO pin, that the level on nDIS is inactive (i.e. high).
- Software sets bit HET PIN ENA to deactivate the high impedance state of the pins.

### 19.2.6 Suppression Filters

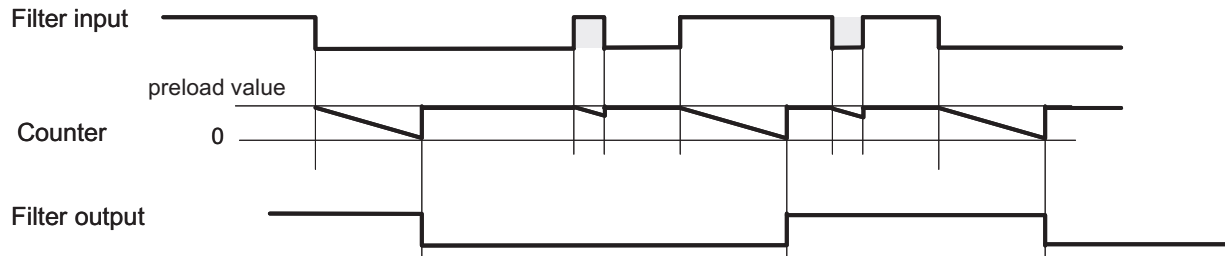
Each N2HET pin is equipped with a suppression filter. If the pin is configured as an input it enables to filter out pulses shorter than a programmable duration. Each filter consists of a 10-bit down counter, which starts counting at a programmable preloaded value and is decremented using the VCLK2 clock.

- The counter starts counting when the filter input signal has the opposite state of the filter output signal. The output signal is preset to the same input signal state after reset, in order to ensure proper operation after device reset.
- Once the counter reaches zero without detecting an opposite pin state on the filter output signal, the output signal is set to the opposite state.
- When the counter detects an opposite pin action on the filter input signal before reaching zero, the counter is loaded with its preload value and the opposite pin action on the filter output signal does not take place. The counter resumes at the preload value until it detects an opposite pin action on the

input signal again.

- Therefore the filter output signal is delayed compared to the filter input signal. The amount of delay depends on the counter clock frequency (VCLK2) and the programmed preload value.
- The accuracy of the output signal is +/- the counter clock frequency.

**Figure 19-26. Suppression Filter Counter Operation**



The following table gives examples for a 100 MHz VCLK2 frequency.

**Table 19-11. Pulse Length Examples for Suppression Filter**

Divider CCDIV	VCLK2	Possible values for the suppressed pulse length / frequency resulting from the programmable 10 bit preload value (0,1,...,1023)	
1	100.0 MHz	10 ns, 20 ns, ..., 10.22 μs, 10.23 μs	50 MHz, 25 MHz, ..., 48.924 kHz, 48.876 kHz
2	50.0 MHz	20 ns, 40 ns, ..., 20.44 μs, 20.48 μs	25 MHz, 12.5 MHz, ..., 24.462 kHz, 24.414 kHz
3	33.3 MHz	30 ns, 60 ns, ..., 30.66 μs, 30.69 μs	16.7 MHz, 8.3 MHz, ..., 16.308 kHz, 16.292 kHz

### 19.2.7 Interrupts and Exceptions

N2HET interrupts can be generated by any instruction that has an interrupt enable bit in its instruction format. When the interrupt condition in an instruction is true and the interrupt enable bit of that instruction is set an interrupt flag is then set in the N2HET Interrupt Flag Register (HETFLG). The address code for this flag is determined by the five LSBs of the current timer program address. The flag in the N2HET Interrupt Flag Register (HETFLG) is set even if the corresponding bit in the N2HET Interrupt Enable Set Register (HETINTENAS) is zero. To generate an interrupt the corresponding bit in the N2HET Interrupt Enable Set Register (HETINTENAS) must be one. In the N2HET interrupt service routine, the main CPU must first determine which source inside the N2HET created the interrupt request. This operation is accelerated by the N2HET Offset Index Priority Level 1 Register (HETOFF1) or N2HET Offset Index Priority Level 2 Register (HETOFF2) which automatically provide the number of the highest priority source within each priority level. Reading the offset register will automatically clear the corresponding N2HET interrupt flag which created the request. However, if the offset registers are not used by the N2HET interrupt service routine, the flag should be cleared explicitly by the CPU once the interrupt has been serviced.

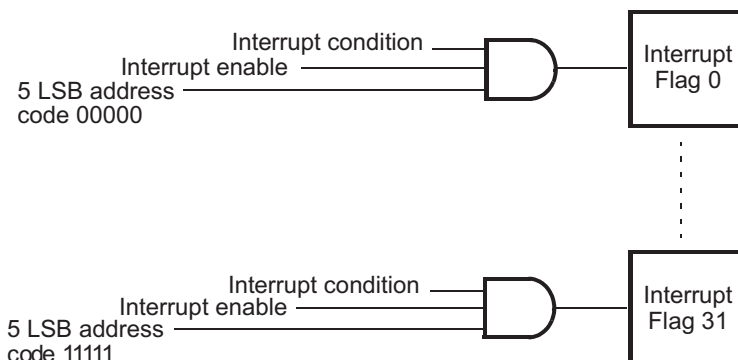
**Table 19-12. Interrupt Sources and Corresponding Offset Values in Registers HETOFFx**

Source No.	Offset Value
no interrupt	0
Instruction 0, 32, 64...	1
Instruction 1, 33, 65...	2
:	:
Instruction 31, 63, 95...	32
Program Overflow	33
APCNT underflow:	34
APCNT overflow	35



The instructions capable of generating interrupts are listed in [Table 19-51](#).

**Figure 19-27. Interrupt Functionality on Instruction Level**



Each interrupt source is associated with a priority level (level 1 or level 2). When multiple interrupts with the same priority level occur during the same loop resolution the lowest flag bit is serviced first.

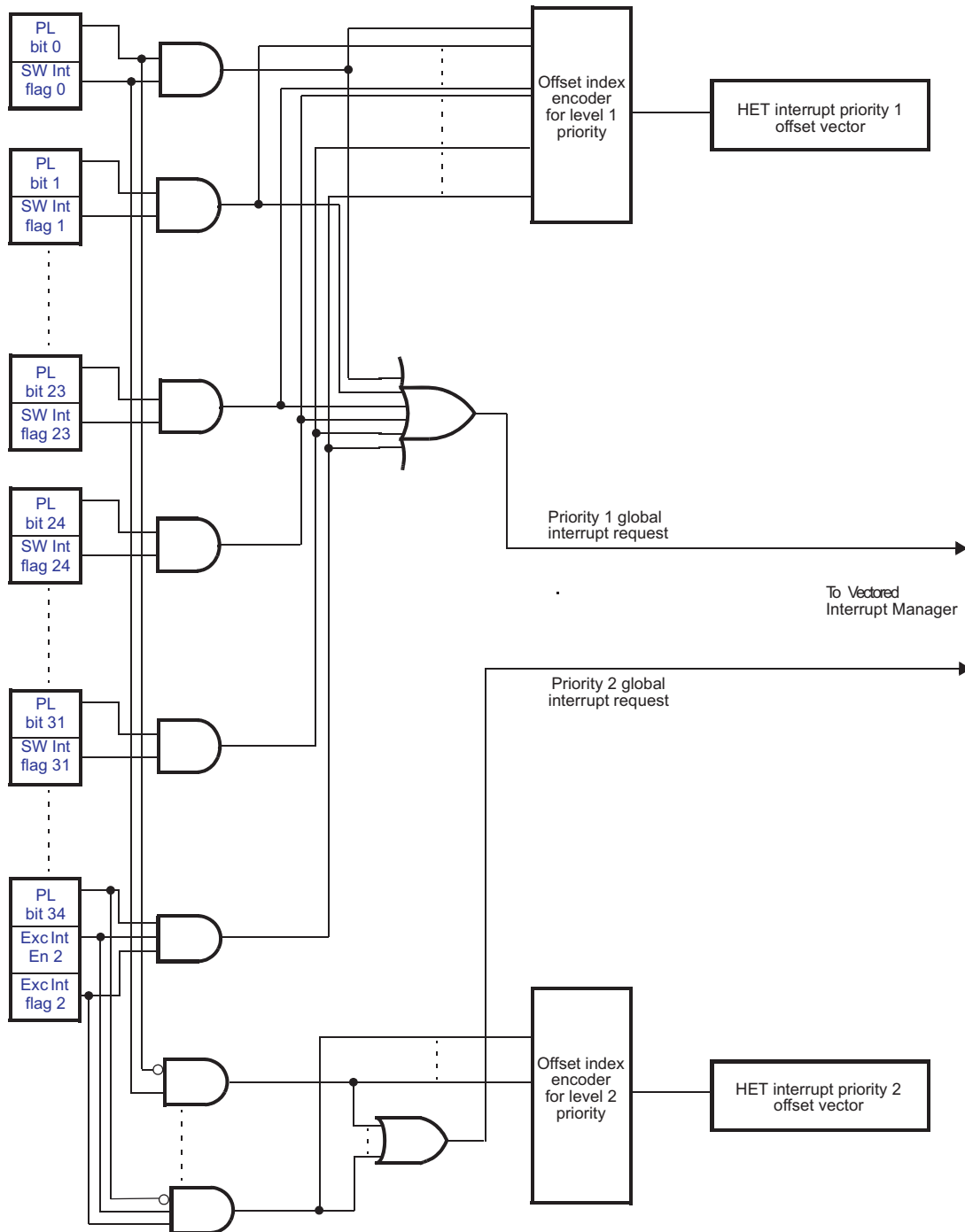
In addition to the interrupts generated by the instructions the N2HET can generate three additional exceptions:

- Program overflow
- APCNT underflow (see [Section 19.3.1.2](#))
- APCNT overflow (see [Section 19.3.1.3](#))

### 19.2.8 Hardware Priority Scheme:

If two or more software interrupts are pending on the same priority level, the offset value will show the one with the highest priority. The interrupt with the highest priority is the one with the lower offset value. This scheme is hard-wired in the offset encoder. See [Figure 19-28](#).

Figure 19-28. Interrupt Flag/Priority Level Architecture

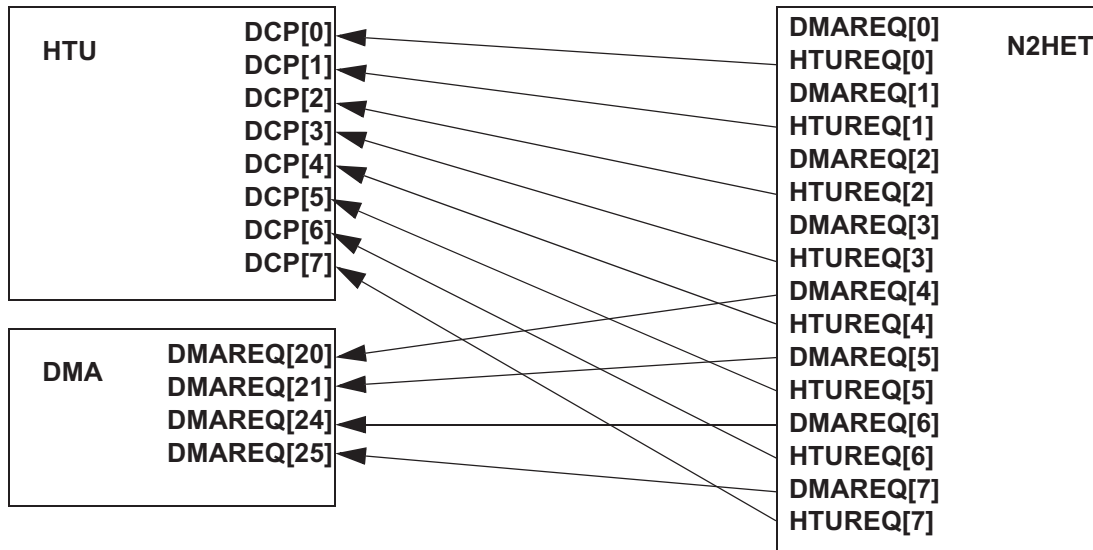


### 19.2.9 N2HET Requests to DMA and HTU

As described in [Section 19.5.3](#) the majority of the N2HET instructions are able to generate a transfer request to the High End Timer Transfer Unit (HTU) and/or to the DMA module when an instruction-specific condition is true. One N2HET instruction can select one of 8 request lines by programming the “reqnum” parameter (see [Section 19.5.3](#)). The “request” field in an instruction is used to enable, disable, or to generate a quiet request (see [Section 19.5.2](#)) on the selected request line. Quiet requests can be used by the HTU, but not by the DMA. For quiet request please refer to the HTU users guide.

The configuration of the N2HET Request Destination Select Register (HETREQDS) bits determines if a request line triggers an HTU-DCP, a DMA channel or both. This means the register bits will determine whether an N2HET instruction triggers DMAREQ[x], HTUREQ[x] or both signals (shown in Figure 19-29). The request line number x corresponds to the “reqnum” parameter used in the instruction.

**Figure 19-29. Request Line Assignment Example**



### 19.3 Angle Functions

Engine management systems require an angle-referenced time base to synchronize signals to the engine toothed wheel. The N2HET has a method to provide such a time base for low-end engine systems. The reference is created by the N2HET using three dedicated instructions with fractional angle steps equal to  $/8$ ,  $/16$ ,  $/32$ ,  $/64$ .

#### 19.3.1 Software Angle Generator

The N2HET provides three specialized count instructions to generate an angle referenced time base synchronized to an external reference signal (the toothed wheel signal) that defines angular reference points.

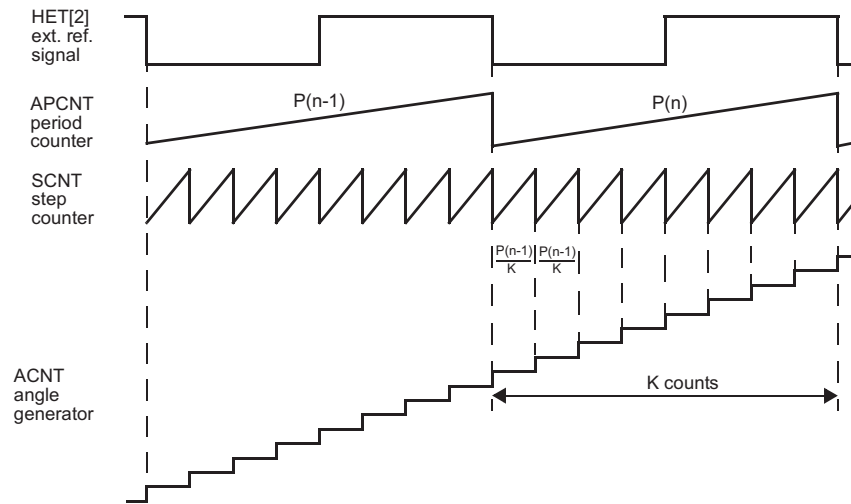
The time base is used to generate fractional angle steps between the reference points. The step width  $K$  ( $= 8, 16, 32, \text{ or } 64$ ) programmed by the user defines the angle accuracy of the time base. These fractional steps are then accumulated in an angle counter to form the absolute angle value.

The first counter, APCNT, incremented on each loop resolution clock measures the periods  $P(n)$  of the external signal. The second counter SCNT counts by step  $K$  up to the previous period value  $P(n-1)$ , measured by APCNT, and then recycles. The resulting period of SCNT is the fraction  $P(n-1) / K$ . The third counter ACNT accumulates the fractions generated by SCNT.

Figure 19-30 illustrates the basic operation of APCNT, SCNT, and ACNT.

A N2HET timer program can only have one angle generator.

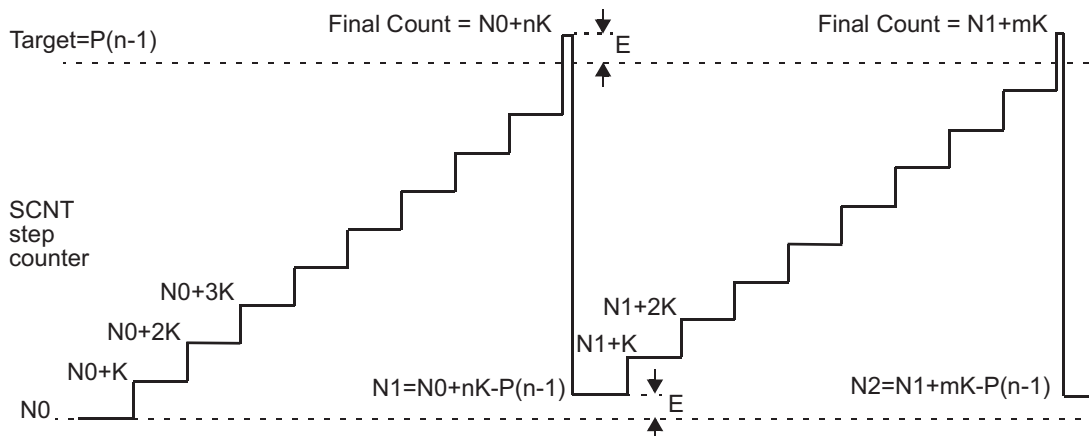
Figure 19-30. Operation of N2HET Count Instructions



Due to stepping, the final count of SCNT does not usually exactly match the target value  $P(n-1)$ .

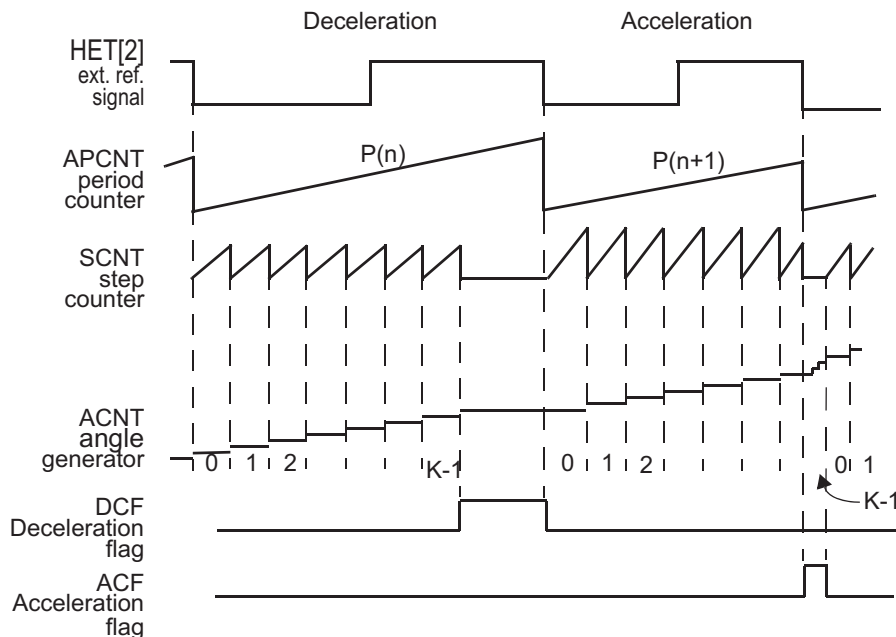
Figure 19-31 illustrates how SCNT compensates for this feature by starting each cycle with the remainder (final count - target) of the previous cycle.

**Figure 19-31. SCNT Count Operation**



ACNT detects period variations of the external signal measured by APCNT and compensates related counting errors. A period increase is flagged in the deceleration flag. A period decrease is flagged in the acceleration flag. If no variation is flagged, ACNT increments the counter value each time SCNT reaches its target. If acceleration is detected, ACNT increments the counter value on each timer resolution (fast mode). If deceleration is detected, ACNT is stopped. Figure 19-32 illustrates how the compensations for acceleration and deceleration operate.

**Figure 19-32. ACNT Period Variation Compensations**



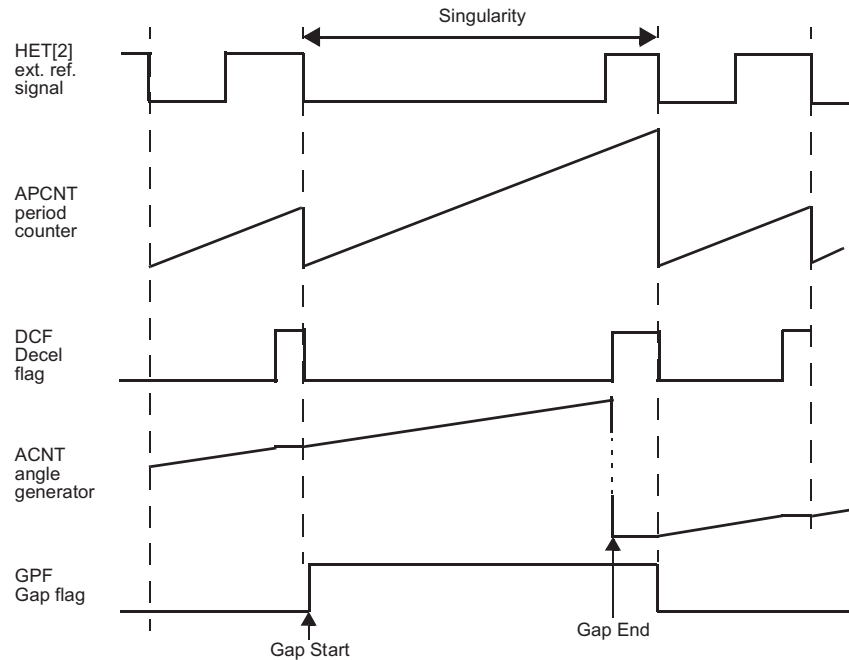
### 19.3.1.1 Singularities

Singularities (gaps, in this case, from missing teeth in a toothed wheel) in the external reference signal can be masked. The start and end of singularities are defined by gap start and gap end values specified in SCNT and ACNT. When ACNT reaches gap start or gap end, it sets/resets the gap flag.

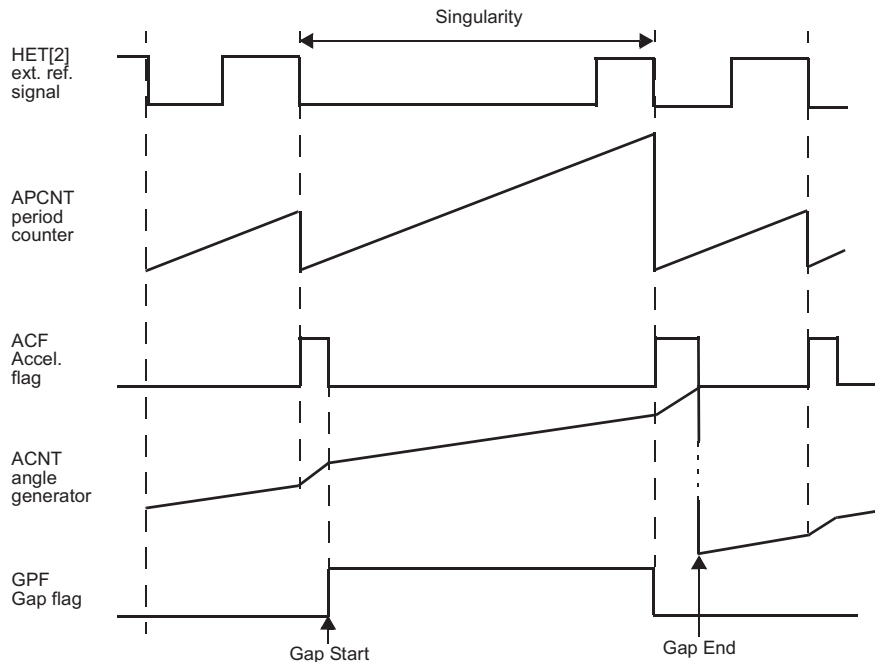
While the gap flag is set, new periods of the external reference signal are ignored for angle computation. SCNT uses the last period measured by APCNT just before gap start.

Figure 19-33 and Figure 19-34 illustrate the behavior of the angle generator during a gap after a deceleration or acceleration of the N2HET.

**Figure 19-33. N2HET Timings Associated with the Gap Flag (ACNT Deceleration)**



**Figure 19-34. N2HET Timings Associated with the Gap Flag (ACNT Acceleration)**



**19.3.1.2 APCNT Underflow**

The fastest valid external signal APCNT can accept must satisfy the following condition:

$$\text{Step Width } K < \text{Period Min. Resolution (LRP)}$$

This condition fixes the maximum possible step width once the minimum period and the resolution of an application are specified.

If a period value accidentally falls below the minimum allowed, APCNT stops the capture of these periods and sets the APCNT underflow interrupt flag located in the exceptions interrupt control register. In such a situation, SCNT and ACNT continue to be executed using the last valid period captured by APCNT.

### 19.3.1.3 APCNT Overflow

The slowest valid external signal APCNT can measure must satisfy the following condition:

Period Max Resolution < 33554431

When this limit is reached (APCNT Count equals all 1's), APCNT stays at a maximum count (stops counting). APCNT remains in this position until the next specified capture edge is detected on the selected pin and sets the APCNT overflow interrupt flag located in the exceptions interrupt control register. In this situation, SCNT and ACNT continue to be executed using the maximum APCNT period count.

## 19.4 N2HET Control Registers

### 19.4.1 N2HET Register Summary

Table 19-13 summarizes all the N2HET registers. The base address for the control registers is FFF7 B800h for N2HET1 and FFF7 B900h for N2HET2.

**Table 19-13. N2HET Registers**

Offset	Acronym	Register Description	Section
00	HETGCR	Global Configuration Register	<a href="#">Section 19.4.2</a>
04h	HETPFR	Prescale Factor Register	<a href="#">Section 19.4.3</a>
08h	HETADDR	NHET Current Address Register	<a href="#">Section 19.4.4</a>
0Ch	HETOFF1	Offset Index Priority Level 1 Register	<a href="#">Section 19.4.5</a>
10h	HETOFF2	Offset Index Priority Level 2 Register	<a href="#">Section 19.4.6</a>
14h	HETINTENAS	Interrupt Enable Set Register	<a href="#">Section 19.4.7</a>
18h	HETINTENAC	Interrupt Enable Clear Register	<a href="#">Section 19.4.8</a>
1Ch	HETEXC1	Exception Control Register 1	<a href="#">Section 19.4.9</a>
20h	HETEXC2	Exception Control Register 2	<a href="#">Section 19.4.10</a>
24h	HETPRY	Interrupt Priority Register	<a href="#">Section 19.4.11</a>
28h	HETFLG	Interrupt Flag Register	<a href="#">Section 19.4.12</a>
2Ch	HETAND	AND Share Control Register	<a href="#">Section 19.4.13</a>
34h	HETHRSH	HR Share Control Register	<a href="#">Section 19.4.14</a>
38h	HETXOR	HR XOR-Share Control Register	<a href="#">Section 19.4.15</a>
3Ch	HETREQENS	Request Enable Set Register	<a href="#">Section 19.4.16</a>
40h	HETREQENC	Request Enable Clear Register	<a href="#">Section 19.4.17</a>
44h	HETREQDS	Request Destination Select Register	<a href="#">Section 19.4.18</a>
4Ch	HETDIR	NHET Direction Register	<a href="#">Section 19.4.19</a>
50h	HETDIN	NHET Data Input Register	<a href="#">Section 19.4.20</a>
54h	HETDOUT	NHET Data Output Register	<a href="#">Section 19.4.21</a>
58h	HETDSET	NHET Data Set Register	<a href="#">Section 19.4.22</a>
5Ch	HETDCLR	NHET Data Clear Register	<a href="#">Section 19.4.23</a>
60h	HETPDR	NHET Open Drain Register	<a href="#">Section 19.4.24</a>
64h	HETPULDIS	NHET Pull Disable Register	<a href="#">Section 19.4.25</a>
68h	HETPSL	NHET Pull Select Register	<a href="#">Section 19.4.26</a>
74h	HETPCR	Parity Control Register	<a href="#">Section 19.4.27</a>
78h	HETPAR	Parity Address Register	<a href="#">Section 19.4.28</a>
7Ch	HETPPR	Parity Pin Register	<a href="#">Section 19.4.29</a>
80h	HETSFPRLD	Suppression Filter Preload Register	<a href="#">Section 19.4.30</a>
84h	HETSFENA	Suppression Filter Enable Register	<a href="#">Section 19.4.31</a>
8Ch	HETLBPEL	Loop Back Pair Select Register	<a href="#">Section 19.4.32</a>
90h	HETLBPDIR	Loop Back Pair Direction Register	<a href="#">Section 19.4.33</a>
94h	HETPINDIS	NHET Pin Disable Register	<a href="#">Section 19.4.34</a>



## 19.4.2 Global Configuration Register (HETGCR)

**Figure 19-35. Global Configuration Register (HETGCR)**

31	25	24	23	22	21	20	19	18	17	16
Reserved		HET PIN ENA	Rsvd.	MP		Reserved		PPF	IS	CMS
R-0		R/W-1	R-0	R/W-0		R-0		R/W-0	R/W-0	R/W-0
15	Reserved								1	0
R-0									TO	R/W-0

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

**Table 19-14. Global Configuration Register (HETGCR)**

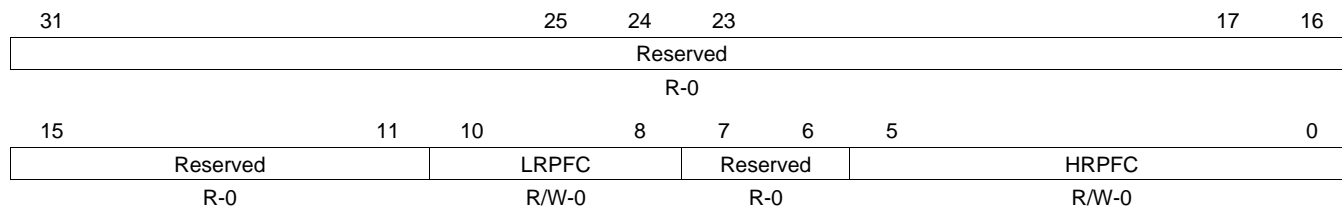
Bit	Field	Value	Description
31-25	Reserved	0	Read returns 0. Writes have no effect.
24	HET PIN ENA	0 1	Enables the output buffers of the pin structures depending on the value of nDIS and DIR.x when PINDIS.x is set. <b>Note:</b> This bit will automatically get cleared when nDIS pin (input port) value is "0". 0 No affect on the pin output buffer structure. 1 Enables the pin output buffer structure when DIR = output, PINDIS.x is set and nDIS = 1.
23	Reserved	0	Read returns 0. Writes have no effect.
22-21	MP	0 1h 2h 3h	Master Priority The NHET can prioritize master accesses to N2HET RAM between the HET Transfer Unit and another arbiter, which outputs the access of one of the remaining masters. The MP bits allow the following selections: 0 The HTU has lower priority to access the N2HET RAM than the arbiter output. 1h The HTU has higher priority to access the N2HET RAM than the arbiter output. 2h The HTU and the arbiter output use a round robin scheme to access the N2HET RAM. 3h Reserved
20-19	Reserved	0	Read returns 0. Writes have no effect.
18	PPF	0 (TO = 0) 0 (TO = 1) 1 (TO = 0) 1 (TO = 1)	Protect Program Fields The PPF bit together with the Turn On/Off bit (TO) allows to protect the program fields of all instructions in N2HET RAM. 0 (TO = 0) All masters can read and write the program fields. 0 (TO = 1) All masters can read and write the program fields. 1 (TO = 0) All masters can read and write the program fields. 1 (TO = 1) The program fields are readable but not writable for all masters, which could access the N2HET RAM. Possible masters are the CPU, HTU, DMA and a secondary CPU (if available). Writes initiated by these masters are discarded.
17	IS	0 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 N2HET continues operating. 0 N2HET stops when in suspend mode. 1 N2HET ignores suspend mode and continues operation.
16	CMS	0 1	Clk_master/slave This bit is used to synchronize multi-N2HETs. If set (N2HET is master), the N2HET outputs a signal to synchronize the prescalers of the slave N2HET. By default, this bit is reset, which means a slave configuration. <b>Note:</b> This bit must be set to one (1) for single-N2HET configuration. 0 N2HET is configured as a slave. 1 N2HET is configured as a master.
15-1	Reserved	0	Read returns 0. Writes have no effect.

**Table 19-14. Global Configuration Register (HETGCR) (continued)**

Bit	Field	Value	Description
0	TO		<p>Turn On/Off</p> <p>When TO = 0, the timer program stops executing. Turn-off is automatically delayed until the current timer program loop is completed. Turn-off does not affect the content of the timer RAM, ALU registers, or control registers. Turn-off resets all flags.</p> <p>TO does not affect the state of the pins. You must set/reset the timer pins when they are turned off, or re-initialize the timer RAM and control registers before a reset. After a device reset, the timer is turned off by default.</p> <p>When TO = 1, timer program execution starts synchronously to the Loop clock. In case of multiple N2HETs configuration, the slave N2HETs are waiting for the loop clock to come from the master before starting execution. Then, the timer address points automatically address 00h (corresponding to program start).</p>
		0	N2HET is OFF.
		1	N2HET is ON.

### 19.4.3 Prescale Factor Register (HETPFR)

**Figure 19-36. Prescale Factor Register (HETPFR)**

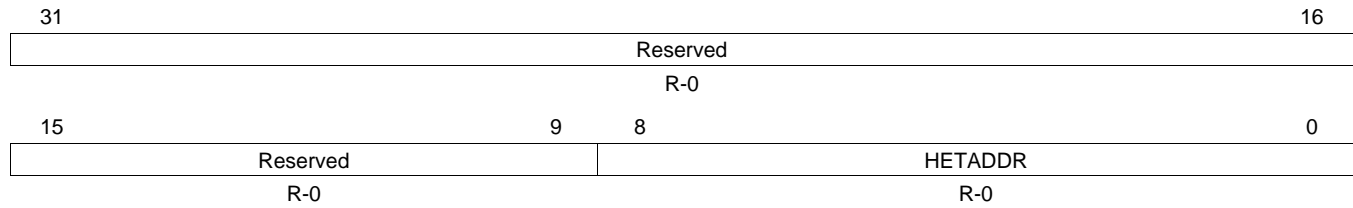


LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-15. Prescale Factor Register (HETPFR)**

Bit	Field	Value	Description
31-11	Reserved	0	Read returns 0. Writes have no effect.
10-8	LRPFC		Loop Resolution Pre-scale Factor Code LRPFC determines the loop resolution prescale divide rate (lr). See <a href="#">Table 19-5</a> .
7-6	Reserved	0	Read returns 0. Writes have no effect.
5-0	HRPFC		High Resolution Pre-scale Factor Code HRPFC determines the high resolution prescale divide rate (hr). See <a href="#">Table 19-5</a> .

### 19.4.4 N2HET Current Address Register (HETADDR)

**Figure 19-37. N2HET Current Address (HETADDR)**


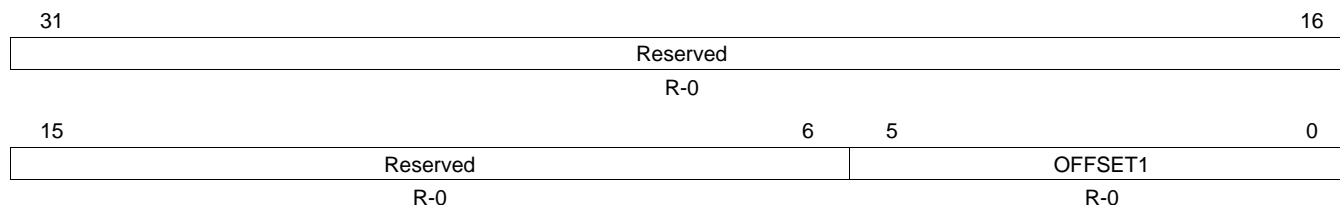
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-16. N2HET Current Address (HETADDR)**

Bit	Field	Value	Description
31-9	Reserved	0	Read returns 0. Writes have no effect.
8-0	HETADDR		N2HET Current Address Read: Returns the current N2HET program address. Write: Writes have no effect.

### 19.4.5 Offset Index Priority Level 1 Register (HETOFF1)

**Figure 19-38. Offset Index Priority Level 1 Register (HETOFF1)**



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

**Table 19-17. Offset Index Priority Level 1 Register (HETOFF1)**

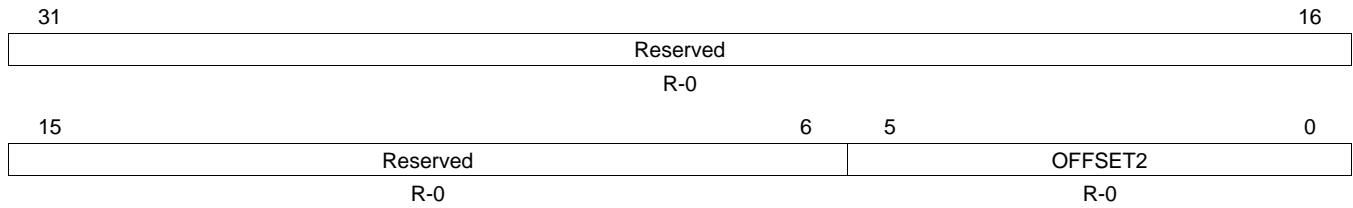
Bit	Field	Value	Description
31-6	Reserved	0	Read returns 0. Writes have no effect.
5-0	OFFSET1		<p>HETOFF1[5:0] indexes the currently pending high-priority interrupt. Offset values and sources are listed below and the interrupt encoding format is presented in <a href="#">Table 19-18</a>.</p> <p>Read: Read of these bits determines the pending N2HET interrupt.</p> <p>Write: Writes have no effect.</p> <p><b>Note:</b> In any read operation mode, the corresponding flag (in the HETFLG) is also cleared. In Emulation mode the corresponding flag is not cleared.</p>

**Table 19-18. 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

**19.4.6 Offset Index Priority Level 2 Register (HETOFF2)**

**Figure 19-39. Offset Index Priority Level 2 Register (HETOFF2) [offset = FFF7 B810h]**



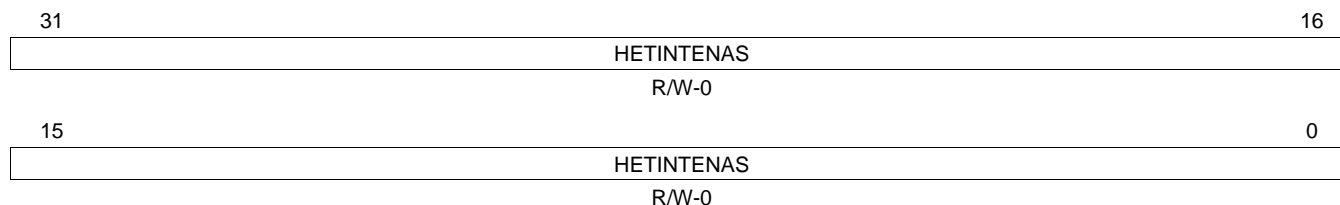
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-19. Offset Index Priority Level 2 Register (HETOFF2)**

Bit	Field	Value	Description
31-6	Reserved	0	Read returns 0. Writes have no effect.
5-0	OFFSET2		HETOFF2[5:0] indexes the currently pending low-priority interrupt. Offset values and sources are listed above and the interrupt encoding format is presented in <a href="#">Table 19-18</a> . Read: Read of these bits determines the pending N2HET interrupt. Write: Writes have no effect. <b>Note:</b> In any read operation mode, the corresponding flag (in the HETFLG) is also cleared. In Emulation mode the corresponding flag is not cleared.

### 19.4.7 Interrupt Enable Set Register (HETINTENAS)

**Figure 19-40. Interrupt Enable Set Register (HETINTENAS)**



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

**Table 19-20. Interrupt Enable Set Register (HETINTENAS)**

Bit	Field	Value	Description
31-0	HETINTENAS		<p>Interrupt Enable Set bits</p> <p>HETINTENAS is readable and writable in any operation mode.</p> <p>Writing a one to bit x enables the interrupts of the N2HET instructions at N2HET addresses x+0, x+32, x+64... Generating an interrupt requires to set bit x in HETINTENAS and to enable the interrupt bit in one of the instructions at addresses x+0, x+32, x+64... To avoid ambiguity only one of the instructions x+0, x+32, x+64... should have the interrupt enable bit (inside the instruction) set. Writing a zero to HETINTENAS has no effect.</p> <p>When reading from HETINTENAS bit x gives the information, if N2HET instructions x+0, x+32, x+64... have the interrupt enabled or disabled.</p>
		0	<p>Read: Interrupt is disabled.</p> <p>Write: Writes have no effect.</p>
		1	<p>Read: Interrupt is enabled.</p> <p>Write: Interrupt will be enabled.</p>

### 19.4.8 Interrupt Enable Clear Register (HETINTENAC)

**Figure 19-41. Interrupt Enable Clear (HETINTENAC)**

31	HETINTENAC	16
R/W-0		
15	HETINTENAC	0
R/W-0		

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

**Table 19-21. NHET Interrupt Enable Clear (HETINTENAC)**

Bit	Field	Value	Description
31-0	HETINTENAC		<p>Interrupt Enable Clear bits</p> <p>HETINTENAC is readable and writable in any operation mode.</p> <p>Writing a one to bit x disables the interrupts of the N2HET instructions at N2HET addresses x+0, x+32, x+64... (See also description in <a href="#">Table 19-20</a>). Writing a zero to HETINTENAC has no effect.</p> <p>When reading from HETINTENAC bit x gives the information, if N2HET instructions x+0, x+32, x+64... have the interrupt enabled or disabled.</p>
		0	<p>Read: Interrupt is disabled.</p> <p>Write: Writes have no effect.</p>
		1	<p>Read: Interrupt is enabled.</p> <p>Write: Interrupt will be enabled.</p>



### 19.4.9 Exception Control Register 1 (HETEXC1)

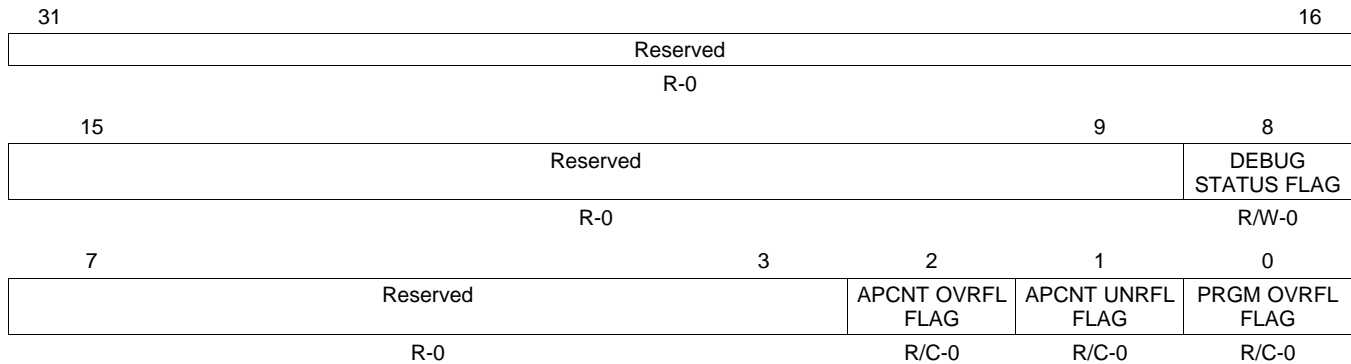
**Figure 19-42. Exception Control Register (HETEXC1)**

31	25	24		
Reserved		APCNT OVRFL ENA		
R-0		R/W-0		
23	17	16		
Reserved		APCNT UNRFL ENA		
R-0		R/W-0		
15	9	8		
Reserved		PRGM OVRFL ENA		
R-0		R/W-0		
7	3	2	1	0
Reserved		APCNT OVRFL PRY	APCNT UNRFL PRY	PRGM OVRFL PRY
R-0		R/W-0	R/W-0	R/W-0

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

**Table 19-22. Exception Control Register 1 (HETEXC1)**

Bit	Field	Value	Description
31-17	Reserved	0	Read returns 0. Writes have no effect.
24	APCNT OVRFL ENA	0	APCNT Overflow Enable APCNT overflow exception is not enabled.
		1	Enables the APCNT overflow exception.
23-17	Reserved	0	Read returns 0. Writes have no effect.
16	APCNT UNRFL ENA	0	APCNT Underflow Enable APCNT underflow exception is not enabled.
		1	Enables the APCNT underflow exception.
15-9	Reserved	0	Read returns 0. Writes have no effect.
8	PRGM OVRFL ENA	0	Program Overflow Enable The program overflow exception is not enabled.
		1	Enables the program overflow exception.
7-3	Reserved	0	Read returns 0. Writes have no effect.
2	APCNT OVRFL PRY	0	APCNT Overflow Exception Interrupt Priority Exception priority level 2
		1	Exception priority level 1
1	APCNT UNRFL PRY	0	APCNT Underflow Exception Interrupt Priority Exception priority level 2
		1	Exception priority level 1
0	PRGM OVRFL PRY	0	ProgramOverflow Exception Interrupt Priority Exception priority level 2
		1	Exception priority level 1

**19.4.10 Exception Control Register 2 (HETEXC2)**
**Figure 19-43. Exception Control Register 2 (HETEXC2)**


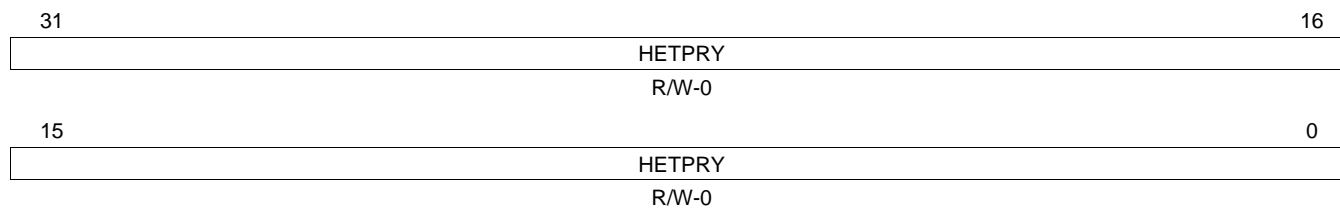
LEGEND: R/W = Read/Write; R = Read only; C = Clear; -n = value after reset

**Table 19-23. Exception Control Register 2 (HETEXC2)**

Bit	Field	Value	Description
31-9	Reserved	0	Read returns 0. Writes have no effect.
8	DEBUG STATUS FLAG	0	Debug Status Flag. This flag is set when N2HET has stopped at a breakpoint. Also generates a debug request to halt the ARM CPU. Read: N2HET is either running, or stopped, flag cleared but not yet restarted. Write: No effect.
		1	Read: N2HET is stopped at a breakpoint. Write: Clears the bit. To restart N2HET clear this bit and then restart the ARM CPU. The N2HET and ARM CPU will start synchronously.
7-3	Reserved	0	Read returns 0. Writes have no effect.
2	APCNT OVRFL FLAG	0	APCNT Overflow Flag Read: Exception has not occurred since the flag was cleared. Write: No effect.
		1	Read: Exception has occurred since the flag was cleared. Write: Clears the bit.
1	APCNT UNDFL FLAG	0	APCNT Underflow Flag Read: Exception has not occurred since the flag was cleared. Write: No effect.
		1	Read: Exception has occurred since the flag was cleared. Write: Clears the bit.
0	PRGM OVERFL FLAG	0	Program Overflow Flag Read: Exception has not occurred since the flag was cleared. Write: No effect.
		1	Read: Exception has occurred since the flag was cleared Write: Clears the bit.

### 19.4.11 Interrupt Priority Register (HETPRY)

**Figure 19-44. Interrupt Priority Register (HETPRY)**

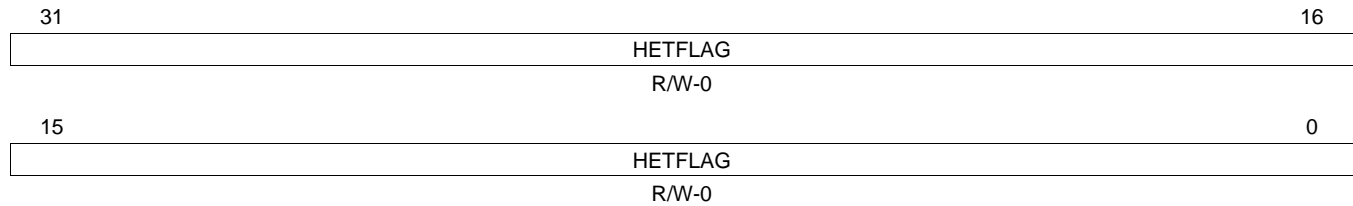


LEGEND: R/W = Read/Write; R = Read only; -n = value after reset; X = Unknown

**Table 19-24. Interrupt Priority Register (HETPRY)**

Bit	Field	Value	Description
31-0	HETPRY		HET Interrupt Priority Level bits Used to select the priority of any of the 32 potential interrupt sources coming from N2HET instructions.
		0	Interrupt priority level 2.
		1	Interrupt priority level 1.

### 19.4.12 Interrupt Flag Register (HETFLG)

**Figure 19-45. Interrupt Flag Register (HETFLG)**


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

**Table 19-25. Interrupt Flag Register (HETFLG)**

Bit	Field	Value	Description
31-0	HETFLAG		<p>Interrupt Flag Register Bits</p> <p>Bit x is set when an interrupt condition has occurred on one of the instructions x+0, x+32, x+64.... The flag position x (in the register) is decoded from the five LSBs of the instruction address that generated the interrupt. The hardware will set the flag only if the interrupt enable bit (in the corresponding instruction) is set. The flag will be set even if bit x in the Interrupt Enable Set Register (HETINTENAS) is not enabled. Enabling bit x in HETINTENAS is required if an interrupt should be generated.</p> <p>Clearing the flag can be done by writing a one to the flag. Alternatively reading the corresponding Offset Index Priority Level 1 Register (HETOFF1) or Offset Index Priority Level 2 Register (HETOFF2) will automatically clear the flag.</p>
		0	<p>Read: No N2HET instruction with an interrupt has been reached since the flag was cleared.</p> <p>Write: No effect.</p>
		1	<p>Read: A N2HET instruction with an interrupt has been reached since the flag was cleared.</p> <p>Write: Clears the bit.</p>

### 19.4.13 AND Share Control Register (HETAND)

**Figure 19-46. AND Share Control Register (HETAND)**

Reserved							
R-0							
15	14	13	12	11	10	9	8
ANDSHARE31/ 30	ANDSHARE29/ 28	ANDSHARE27/ 26	ANDSHARE25/ 24	ANDSHARE23/ 22	ANDSHARE21/ 20	ANDSHARE19/ 18	ANDSHARE17/ 16
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
ANDSHARE15/ 14	ANDSHARE13/ 12	ANDSHARE11/ 10	ANDSHARE9/8	ANDSHARE7/6	ANDSHARE5/4	ANDSHARE3/2	ANDSHARE1/0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

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

**Table 19-26. AND Share Control Register (HETAND)**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	AND SHARE n+1 / n	0	AND Share Enable Enable the AND sharing of the same pin for two HR structures. For example, if bit AND SHARE 1/0 is set, the pin HET[0] will then be commanded by a logical AND of both HR structures 0 and 1. <b>Note:</b> If HR AND SHARE bits are used, pins not connected to HR structures (the odd number pin in each pair) can be accessed as general inputs/outputs.
		1	HR Output of HET[n+1] and HET[n] are not AND shared
		1	HR Output of HET[n+1] and HET[n] are AND shared onto pin HET[n]

**19.4.14 HR Share Control Register (HETHRSH)**
**Figure 19-47. HR Share Control Register (HETHRSH)**

31								16							
Reserved															
R-0															
15		14		13		12		11		10		9		8	
HRSHARE31/3 0	HRSHARE29/2 8	HRSHARE27/2 6	HRSHARE25/2 4	HRSHARE23/2 2	HRSHARE21/2 0	HRSHARE19/1 8	HRSHARE17/1 6	HRSHARE15/1 4	HRSHARE13/1 2	HRSHARE11/1 0	HRSHARE9/8	HRSHARE7/6	HRSHARE5/4	HRSHARE3/2	HRSHARE1/0
R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0	
7		6		5		4		3		2		1		0	
HRSHARE15/1 4	HRSHARE13/1 2	HRSHARE11/1 0	HRSHARE9/8	HRSHARE7/6	HRSHARE5/4	HRSHARE3/2	HRSHARE1/0	HRSHARE15/1 4	HRSHARE13/1 2	HRSHARE11/1 0	HRSHARE9/8	HRSHARE7/6	HRSHARE5/4	HRSHARE3/2	HRSHARE1/0
R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0	

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

**Table 19-27. HR Share Control Register (HETHRSH)**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	HR SHARE n+1 / n	0  1	<p>HR Share Bits</p> <p>Enables the share of the same pin for two HR structures. For example, if bit HR share 1/0 is set, the pin HET[0] will then be connected to both HR input structures 0 and 1.</p> <p><b>Note:</b> If HR share bits are used, pins not connected to HR structures (the odd number pin in each pair) can be accessed as general inputs/outputs.</p> <p>0 HR Input of HET[n+1] and HET[n] are not shared.</p> <p>1 HR Input of HET[n+1] and HET[n] are shared; both measure pin HET[n]</p>

### 19.4.15 XOR Share Control Register (HETXOR)

**Figure 19-48. XOR Share Control Register (HETXOR)**

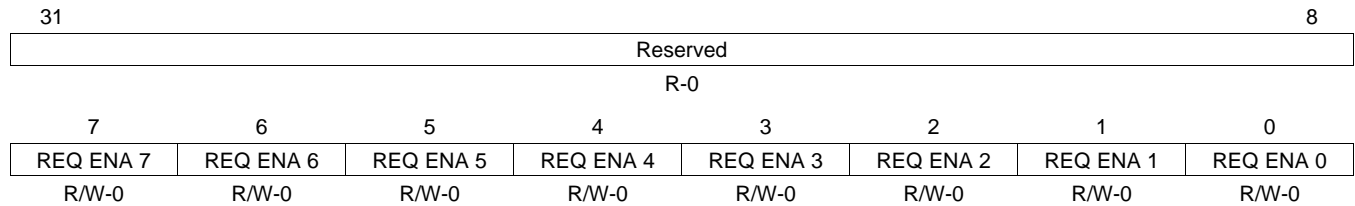
Reserved								
R-0								
31	15	14	13	12	11	10	9	8
	XORSHARE31/ 30	XORSHARE29/ 28	XORSHARE27/ 26	XORSHARE25/ 24	XORSHARE23/ 22	XORSHARE21/ 20	XORSHARE19/ 18	XORSHARE17/ 16
	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	7	6	5	4	3	2	1	0
	XORSHARE15/ 14	XORSHARE13/ 12	XORSHARE11/ 10	XORSHARE9/8	XORSHARE7/6	XORSHARE5/4	XORSHARE3/2	XORSHARE1/0
	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

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

**Table 19-28. XOR Share Control Register (HETXOR)**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	XOR SHARE n+1 / n	0	XOR Share Enable Enable the XOR-share of the same pin for two output HR structures. For example, if bit XOR SHARE 1/0 is set, the pin HET[0] will then be commanded by a logical XOR of both HR structures 0 and 1. <b>Note:</b> If XOR share bits are used, pins not connected to HR structures (the odd number pin in each pair) can be accessed as general inputs/outputs.
		1	HR Output of HET[n+1] and HET[n] are not XOR shared
		1	HR Output of HET[n+1] and HET[n] are XOR shared onto pin HET[n]

### 19.4.16 Request Enable Set Register (HETREQENS)

**Figure 19-49. Request Enable Set Register (HETREQENS)**


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

**Table 19-29. Request Enable Set Register (HETREQENS)**

Bit	Field	Value	Description
31-8	Reserved	0	Read returns 0. Writes have no effect.
7-0	REQ ENA n	0	Request Enable Bits Read: Returns the information that request line n is disabled. Write: Writing a 0 has no effect.
		1	Read: Returns the information that request line n is enabled. Write: Writing a 1 to bit n enables the N2HET request line n. <b>Note:</b> The request line can trigger a DMA control packet (i.e. DMA channel), an HTU double control packet (DCP) or both simultaneously. The HETREQDS register determines to which module(s) the N2HET request line n is assigned. <b>Note:</b> A disabled request line does not memorize old requests. So there are no pending requests to service after enabling request line n.





### 19.4.18 Request Destination Select Register (HETREQDS)

**Figure 19-51. Request Destination Select Register (HETREQDS) [offset = FFF7 B844h]**

31	24	23								16
Reserved		TDBS 7	TDBS 6	TDBS 5	TDBS 4	TDBS 3	TDBS 2	TDBS 1	TDBS 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	
15	8	7								0
Reserved		TDS 7	TDS 6	TDS 5	TDS 4	TDS 3	TDS 2	TDS 1	TDS 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	

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

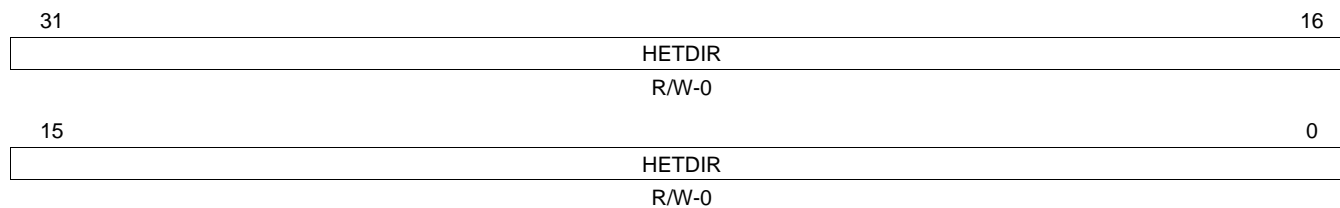
**Table 19-31. Request Destination Select Register (HETREQDS)**

Bit	Field	Value	Description
31-24	Reserved	0	Read returns 0. Writes have no effect.
23-16	TDBS n	0	N2HET request line n is assigned to the module specified by TDS bit n.
		1	N2HET request line n is assigned to both DMA and HTU. TDS bit n is ignored in this case.
15-8	Reserved	0	Read returns 0. Writes have no effect.
7-0	TDS n		HTU or DMA Select Bits <b>Note:</b> It must be ensured in the N2HET program, that one request line is triggered by only one N2HET instruction.
		0	N2HET request line n is assigned to HTU (TDBS bit n is zero).
		1	N2HET request line n is assigned to DMA (TDBS bit n is zero).

**NOTE:** Please refer to the device data sheet how each of the 8 N2HET request lines are connected to these modules. See also [Section 19.2.9](#).

### 19.4.19 NHET Direction Register (HETDIR)

**Figure 19-52. N2HET Direction Register (HETDIR)**

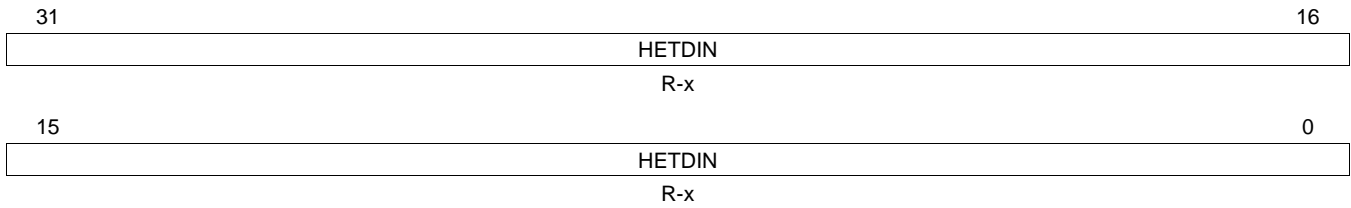


LEGEND: R/W = Read/Write; R = Read only; -n = value after reset; X = Unknown

**Table 19-32. N2HET Direction Register (HETDIR)**

Bit	Field	Value	Description
31-0	HETDIR n	0	Data direction of NHET pins Pin HET[n] is an input (and its output buffer is tristated)
		1	Pin HET[n] is an output

**NOTE:** [Table 19-9](#) shows how the register bits of DIR, PULDIS and PULSEL are affecting the N2HET pins.

**19.4.20 N2HET Data Input Register (HETDIN)**
**Figure 19-53. N2HET Data Input Register (HETDIN)**


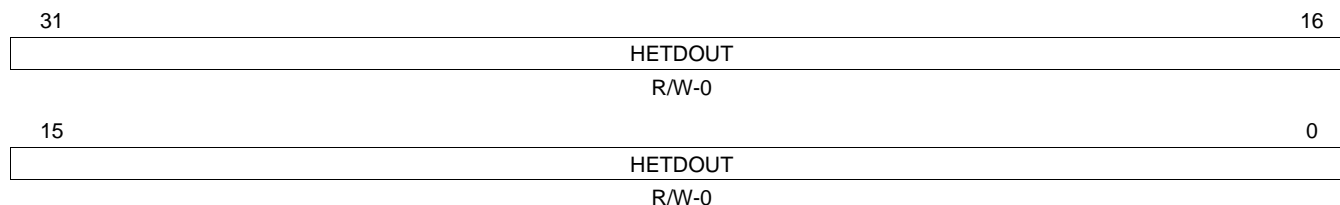
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-33. N2HET Data Input Register (HETDIN)**

Bit	Field	Value	Description
31-0	HETDIN n	0	Data input. This bit displays the logic state of the pin. Pin HET[n] is at logic low (0)
		1	Pin HET[n] is at logic high (1)

### 19.4.21 N2HET Data Output Register (HETDOUT)

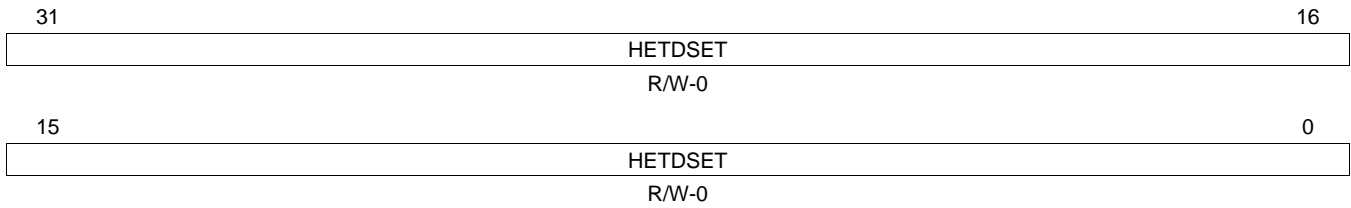
**Figure 19-54. N2HET Data Output Register (HETDOUT)**



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

**Table 19-34. N2HET Data Output Register (R-Write) (HETDOUT)**

Bit	Field	Value	Description
31-0	HETDOUT n		Data out write. Writes to this bit will only take effect when the pin is configured as an output. The current logic state of the pin will be displayed by this bit even when the pin state is changed by writing to HETDSET or HETDCLR.
		0	Pin HET[n] is at logic low (0).
		1	Pin HET[n] is at logic high (1) if the HETPDR[n] bit = 0 or the output is in high impedance state if the HETPDR[n] bit = 1

**19.4.22 NHET Data Set Register (HETDSET)**
**Figure 19-55. N2HET Data Set Register (HETDSET)**


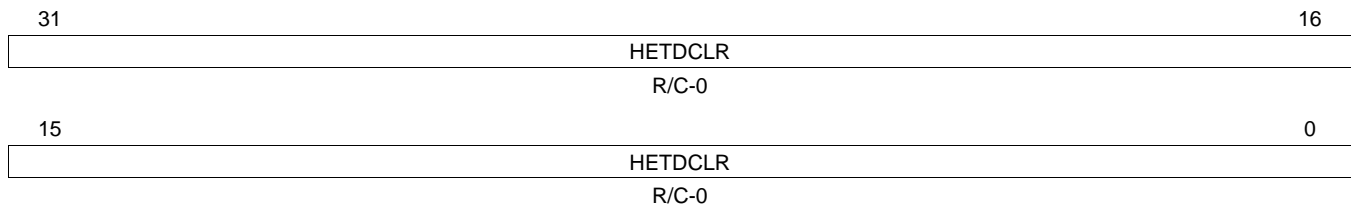
LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

**Table 19-35. N2HET Data Set Register (HETDSET)**

Bit	Field	Value	Description
31-0	HETDSET n		This register allows bits of HETDOUT to be set while avoiding the pitfalls of a readmodify- write sequence in a multitasking environment.
		0	Write: HETDOUT[n] unchanged.
		1	Write: HETDOUT[n] is set.

### 19.4.23 N2HET Data Clear Register (HETDCLR)

**Figure 19-56. N2HET Data Clear Register (HETDCLR)**



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

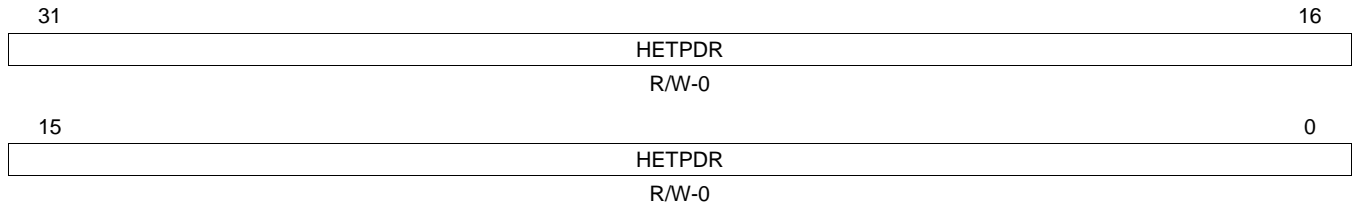
**Table 19-36. N2HET Data Clear Register (HETDCLR)**

Bit	Field	Value	Description
31-0	HETDCLR n		This register allows bits of HETDOUT to be cleared while avoiding the pitfalls of a read-modify-write sequence in a multitasking environment.  Bits written as a logic 1 clear the same bit in the HETDOUT register; while bits written as logic 0 leave the same bit in HETDOUT unchanged. Reads from this address return the value of the HETDOUT register.
		1	Write: HETDOUT[n] unchanged.
		0	Write: HETDOUT[n] is cleared.

### 19.4.24 N2HET Open Drain Register (HETPDR)

Values in this register enable or disable the open drain capability of the data pins.

**Figure 19-57. N2HET Open Drain Register (HETPDR)**



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

**Table 19-37. N2HET Open Drain Register (HETPDR)**

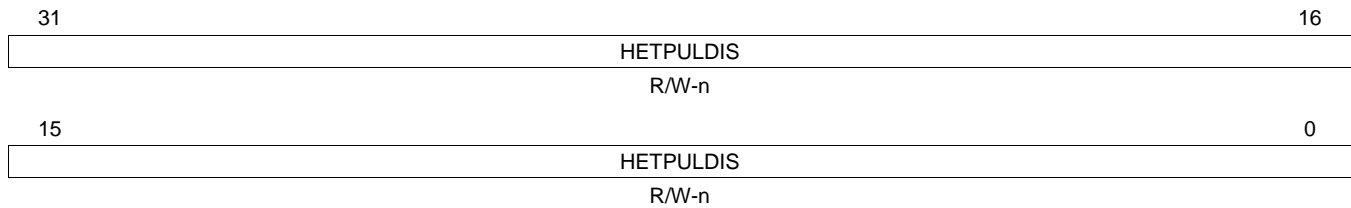
Bit	Field	Value	Description
31-0	HETPDR n	0	Open drain control for HET[n] pins The pin is configured in push/pull mode.
		1	The pin is configured in open drain mode. The HETDOUT register controls the state of the output buffer: HETDOUT[n] = 0 The output buffer of pin HET[n] is driven low. HETDOUT[n] = 1 The output buffer of pin HET[n] is tristated.



### 19.4.25 N2HET Pull Disable Register (HETPULDIS)

Values in this register enable or disable the pull-up/-down functionality of the pins.

**Figure 19-58. N2HET Pull Disable Register (HETPULDIS)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset; n is device dependent, see device specific data sheet

**Table 19-38. N2HET Pull Disable Register (HETPULDIS)**

Bit	Field	Value	Description
31-0	HETPULDIS n	0	Pull disable for N2HET pins The pull functionality is enabled on pin HET[n].
		1	The pull functionality is disabled on pin HET[n].

---

**NOTE:** See device data sheet which pins provide programmable pullups/pulldowns.

---



---

**NOTE:** [Table 19-9](#) shows how the register bits of HETDIR, HETPULDIS and HETPSL are affecting the N2HET pins.

---

### 19.4.26 N2HET Pull Select Register (HETPSL)

Values in this register select the pull-up or pull-down functionality of the pins.

**Figure 19-59. N2HET Pull Select Register (HETPSL)**

31	HETPSL R/W-0	16
15	HETPSL R/W-0	0

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

**Table 19-39. N2HET Pull Select Register (HETPSL)**

Bit	Field	Value	Description
31-0	HETPSL n	0	Pull select for NHET pins The pull down functionality is enabled if corresponding bit in HETPULDIS is 0.
		1	The pull up functionality is enabled if corresponding bit in HETPULDIS is 0.

---

**NOTE:** See device data sheet which pins provide programmable pullups/pulldowns.

---

**NOTE:** [Table 19-9](#) shows how the register bits of HETDIR, HETPULDIS and HETPSL are affecting the N2HET pins.

---

**NOTE:** The information of this register is also used to define the pin states after a parity error:

After a parity error all N2HET pins, which are

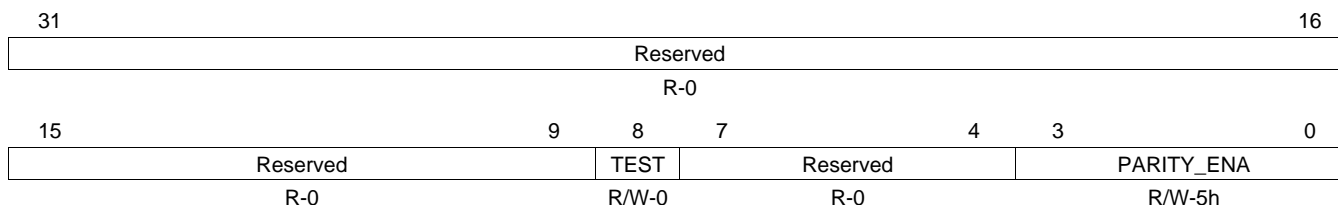
1. Defined as output pins in the HETDIR register
2. Not defined as open drain pins (with the HETPDR register)
3. Selected with the HETPPR register, will remain outputs, but automatically change their levels in the following way:
  - If the HETPSL register specifies 0 for the pin, it will switch to low level.
  - If the HETPSL register specifies 1 for the pin, it will switch to high level.

This behavior is independent of the value, which register HETPULDIS specifies for the corresponding pin.

---

### 19.4.27 Parity Control Register (HETPCR)

**Figure 19-60. Parity Control Register (HETPCR)**

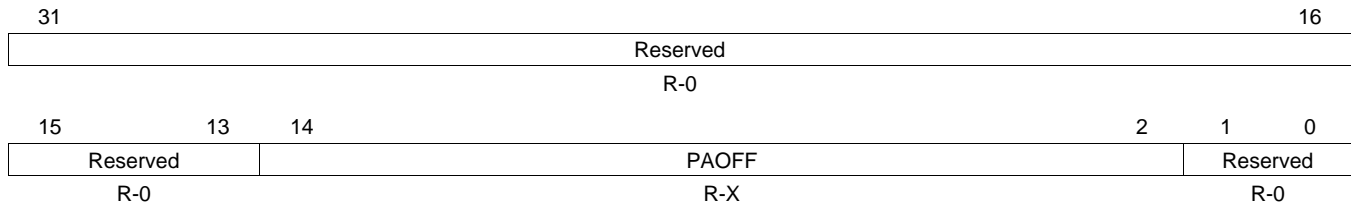


LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-40. Parity Control Register (HETPCR)**

Bit	Field	Value	Description
31-9	Reserved	0	Read returns 0. Writes have no effect.
8	TEST	0	Test Bit. When this bit is set, the parity bits are mapped into the peripheral RAM frame to make them accessible by the CPU. Read: Parity bits are not memory mapped. Write: Disable mapping.
		1	Read: Parity bits are memory mapped. Write: Enable mapping.
7-4	Reserved	0	Read returns 0. Writes have no effect.
3-0	PARITY_ENA	5h	Enable/disable parity checking. This bit field enables or disables the parity check on read operations and the parity calculation on write operations. If parity checking is enabled and a parity error is detected the N2HET_UERR signal is activated. Read: Parity check is disabled. Write: Disable checking.
		Others	Read: Parity check is enabled. Write: Enable checking.

**NOTE:** It is recommended to write an "Ah" to enable error detection, to guard against soft errors flipping PARITY\_ENA to a disable state.

**19.4.28 Parity Address Register (HETPAR)**
**Figure 19-61. Parity Address Register (HETPAR)**


LEGEND: R/W = Read/Write; R = Read only; -n = value after reset; X = Value unchanged after reset

**Table 19-41. Parity Address Register (HETPAR)**

Bit	Field	Value	Description
31-13	Reserved	0	Read returns 0. Writes have no effect.
12-2	PAOFF		<p>Parity Error Address Offset. This register holds the offset address of the first parity error, which is detected in N2HET RAM. This error address is frozen from being updated until it is read by the CPU. During emulation mode, this address is frozen even when read.</p> <p>In case of a N2HET RAM parity error, PAOFF will contain the offset address of the erroneous 32-bit N2HET RAM field counted from the beginning of the N2HET RAM.</p> <p>Examples: The 32-bit program field of instruction 0 will return 0, the 32-bit control field of instruction 0 will return 1, ..., the 32-bit control field of instruction 1 will return 5, and so on.</p> <p>Read: Returns the offset address of the erroneous 32-bit word in bytes from the beginning of the N2HET RAM.</p> <p>Write: Writes have no effect.</p>
1-0	Reserved	0	Read returns 0. Writes have no effect.

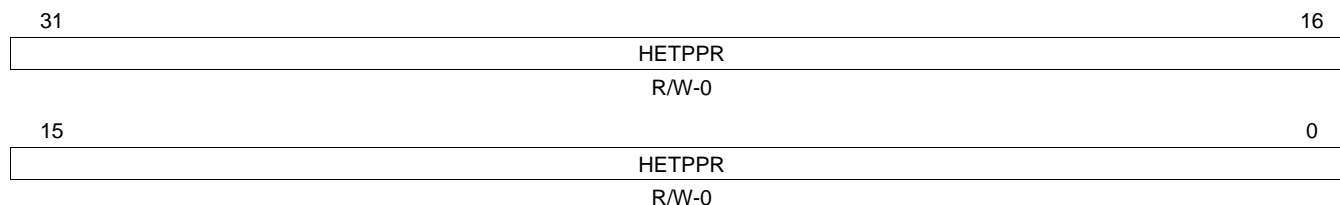
---

**NOTE:** The Parity Error Address Register will not be reset, neither by PORRST nor by any other reset source.

---

### 19.4.29 Parity Pin Register (HETPPR)

**Figure 19-62. Parity Pin Register (HETPPR)**



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

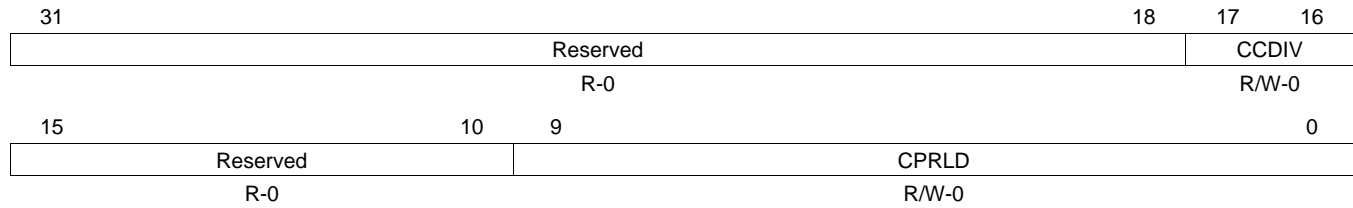
**Table 19-42. Parity Pin Register (HETPPR)**

Bit	Field	Value	Description
31-0	HETPPR n	0	NHET Parity Pin Select Bits - Allows HET[n] pins to be configured to drive to a known state when an N2HET parity error is detected. Pin HET[n] is not affected by the detection of an N2HET parity error.
		1	Pin HET[n] is driven to a known state when an N2HET parity error is detected. The known state is a function of bits HETDIR[n], HETPSL[n], HETPDR[n] as described in <a href="#">Table 19-43</a> . (This state is also independent of HETPULDIS[n]).

**Table 19-43. Known State on Parity Error**

HETDIR[n]	HETPDR[n]	HETPSL[n]	Known State on Parity Error
0	x	x	High Impedance
1	0	0	Drive Logic 0
1	0	1	Drive Logic 1
1	1	x	High Impedance

### 19.4.30 Suppression Filter Preload Register (HETSFPRLD)

**Figure 19-63. Suppression Filter Preload Register (HETSFPRLD)**


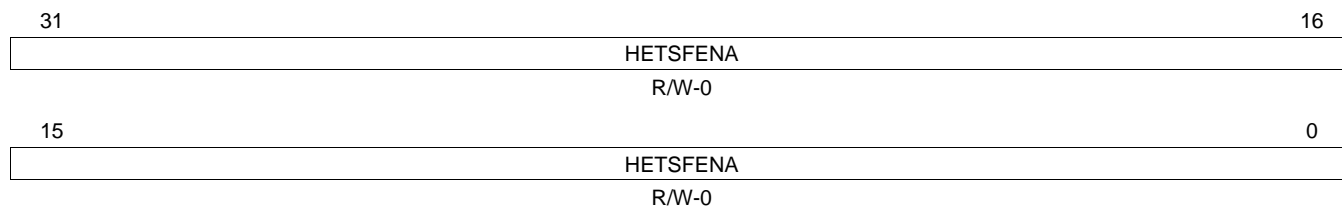
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-44. Suppression Filter Preload Register (HETSFPRLD)**

Bit	Field	Value	Description
31-18	Reserved	0	Read returns 0. Writes have no effect.
17-16	CCDIV	0	Counter Clock Divider CCDIV determines the ratio between the counter clock and VCLK2.
		1h	CCLK = VCLK2 / 2
		2h	CCLK = VCLK2 / 3
		3h	CCLK = VCLK2 / 4
15-10	Reserved	0	Read returns 0. Writes have no effect.
9-0	CPRLD		Counter Preload Value CPRLD contains the preload value for the counter clock.

### 19.4.31 Suppression Filter Enable Register (HETSFENA)

**Figure 19-64. Suppression Filter Enable Register (HETSFENA)**



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

**Table 19-45. Suppression Filter Enable Register (HETSFENA)**

Bit	Field	Value	Description
31-0	HETSFENA n		Suppression Filter Enable Bits <b>Note:</b> If the pin is configured as an output by the N2HET Direction Register (HETDIR), the filter is automatically disabled independent on the bit in HETSFENA.
		0	The input noise suppression filter for pin HET[n] is disabled.
		1	The input noise suppression filter for pin HET[n] is enabled.

**19.4.32 Loop Back Pair Select Register (HETLBPSEL)**
**Figure 19-65. Loop Back Pair Select Register (HETLBPSEL)**

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

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

**Table 19-46. Loop Back Pair Select Register (HETLBPSEL)**

Bit	Field	Value	Description
31-16	LBPTYPE n+1 / n	0 1	<p>Loop Back Pair Type Select Bits</p> <p>These bits are valid only when IODFT mode is enabled (HETLBPDIR[19:16] = "1010").</p> <p>Digital loopback is selected for HR structures on pins HET[n+1] and HET[n].</p> <p>Analog loopback is selected for HR structures on pins HET[n+1] and HET[n].</p>
15-0	LBPSEL n+1 / n		<p>Loop Back Pair Select Bits</p> <p>These bits are valid only when IODFT mode is enabled (HETLBPDIR[19:16] = "Ah").</p> <p>If bit x is set, the HR structures on pins HET[n+1] and HET[n] are connected in a loop back mode. The direction is given by LBPDIR n+1/n and type is selected by LBPTYPE n+1/n.</p> <p>The pin which is not driven by the N2HET pin actions can still be used as normal GIO pin.</p>



### 19.4.33 Loop Back Pair Direction Register (HETLBPDIR)

**Figure 19-66. Loop Back Pair Direction Register (HETLBPDIR)**

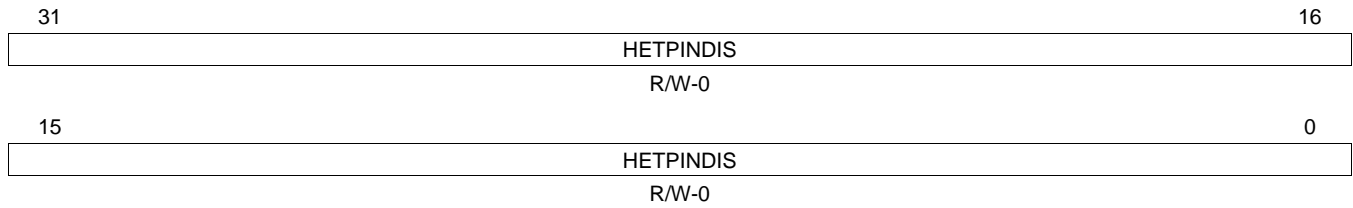
31								20		19		16			
Reserved								IODFTENA							
R-0								R/W-5h							
15		14		13		12		11		10		9		8	
LBP DIR 31/30	LBP DIR 29/28	LBP DIR 27/26	LBP DIR 25/24	LBP DIR 23/22	LBP DIR 21/20	LBP DIR 19/18	LBP DIR 17/16								
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0								
7		6		5		4		3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

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

**Table 19-47. Loop Back Pair Direction Register (HETLBPDIR)**

Bit	Field	Value	Description
31-20	Reserved	0	Read returns 0. Writes have no effect.
19-16	IODFTENA	Ah Others	Module IODFT Enable Key Reset Value is "0101" i.e. I/O DFT is enabled. I/O DFT is enabled. I/O DFT is disabled.
15-0	LBPDIR n+1 / n	0 1	Loop Back Pair Direction Bits The HR structures on pins HET[n+1] and HET[n] are internally connected with HET[n] as input and HET[n+1] as output. The HR structures on pins HET[n+1] and HET[n] connected with HET[n] as output and HET[n+1] as input.

**NOTE:** The loop back direction can be selected independent on the HETDIR register setting.

**19.4.34 N2HET Pin Disable Register (HETPINDIS)**
**Figure 19-67. N2HET Pin Disable Register (HETPINDIS)**


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

**Table 19-48. NHET Pin Disable Register (HETPINDIS)**

Bit	Field	Value	Description
31-0	HETPINDIS n	0	Logic low: No affect on the output buffer enable of the pin (is controlled by the value of the HETDIR[n] bit).
		1	Logic high: Output buffer of the pin is enabled if pin nDIS = 1, HET PIN ENA = 1 and HETDIR = 1, or else disabled if nDIS = 0 or HETDIR = 0 or HET PIN ENA = 0.

## 19.5 Instruction Set

### 19.5.1 Instruction Summary

Table 19-49 presents a list of the instructions in the N2HET instruction set. The pages following describe each instruction in detail.

**Table 19-49. Instruction Summary**

Abbreviation	Instruction Name	Opcode	Sub-Opcode	Cycles
ACMP	Angle Compare	Ch	-	1
ACNT	Angle Count	9h	-	2
ADCNST	Add Constant	5h	-	2
ADC	Add with Carry and Shift	4h	C[25:23]=001, C5 = 1	1 - 3
ADD	Add and Shift	4h	C[25:23]=011, C5 = 1	1 - 3
ADM32	Add Move 32	4h	C[25:23]=000, C5 = 1	1 - 2
AND	Bitwise AND and Shift	4h	C[25:23]=010, C5 = 1	1 - 3
APCNT	Angle Period Count	Eh	-	1 - 2
BR	Branch	Dh	-	1
CNT	Count	6h	-	1 - 2
DADM64	Data Add Move 64	2h	-	2
DJZ	Decrement and Jump if -zero	Ah	P[7:6] = 10	1
ECMP	Equality Compare	0h	C[6:5] = 00	1
ECNT	Event Count	Ah	P[7:6] = 01	1
MCMP	Magnitude Compare	0h	C[6] = 1	1
MOV32	Move 32	4h	C[5] = 0	1 - 2
MOV64	Move 64	1h	-	1
PCNT	Pulse/Period Count	7h	-	1
PWCNT	Pulse Width Count	Ah	P[7:6]=11	1
RADM64	Register Add Move 64	3h	-	1
RCNT	Ratio Count	Ah	P[7:6]=00, P[0]=1	3
SBB	Subtract with Borrow and Shift	4h	C[25:23] =110 C[5] =1	1 - 3
SCMP	Sequence Compare	0h	C[6:5] = 01	1
SCNT	Step Count	Ah	P[7:6] = 00, P[0] = 0	3
SHFT	Shift	Fh	C[3]=0	1
SUB	Subtract and Shift	4h	C[25:23]=101, C[5] = 1	1 - 3
WCAP	Software Capture Word	Bh	-	1
WCAPE	Software Capture Word and Event Count	8h	-	1
XOR	Bitwise Exclusive-Or and Shift	4h	C[25:23] , C[5] = 1	1 - 3

**Table 19-50. FLAGS Generated by Instruction**

Abbreviation	Flag Name	Set/Reset by	Used by
C	Carry Flag	ADC, ADD, AND, OR, RCNT, SBB, SUB, XOR	ADC, BR, SBB
N	Negative Flag	ADC, ADD, AND, OR, SBB, SUB, XOR	BR
V	Overflow Flag	ADC, ADD, AND, OR, SBB, SUB, XOR	BR
Z	Zero flag	ACNT, ADC, ADD, AND, APCNT, CNT, OR, PCNT, SBB, SCNT, SHFT, XOR,	ACMP, ACNT, BR, ECMP, MCMP, MOV32, RCNT, SCMP, SHFT
X	Angle Compare Match Flag	ACMP	SCMP
SWF 0-1	Step width flags	SCNT	ACNT

**Table 19-50. FLAGS Generated by Instruction (continued)**

Abbreviation	Flag Name	Set/Reset by	Used by
NAF	New Angle Flag	ACNT	NAF_global
NAF_global	New Angle flag (global)	<i>HWAG or NAF</i>	ACMP, BR, CNT, ECMP, ECNT
ACF	Acceleration flag	ACNT	,ACNT, SCNT
DCF	Deceleration flag	ACNT	,ACNT, SCNT
GPF	Gap flag	ACNT	ACNT, APCNT

The instructions capable of generating SW interrupts are listed in [Table 19-51](#).

**Table 19-51. Interrupt Capable Instructions**

Interrupt Capable Instructions	Non Interrupt Capable Instructions
ACMP	ADC
ACNT	ADCNST
APCNT	ADD
BR	ADM32
CNT	AND
DJZ	DADM32
ECMP	MOV32
ECNT	MOV64
MCMP	OR
PCNT	RADM64
PWCNT	RCNT
SCMP	SBB
SHFT	SCNT
WCAP	SUB
WCAPE	XOR

### 19.5.2 Abbreviations, Encoding Formats and Bits

Abbreviations marked with a star (\*) are available only on specific instructions.

<b>U</b>	Reading a bit marked with U will return an indeterminate value.
<b>BRK</b>	Defines the software breakpoint for the device software debugger. Default: OFF Location: Program field [22]
<b>next</b>	Defines the program address of the next instruction in the program flow. This value may be a label or an 9-bit unsigned integer. Default: Current instruction + 1 Location: Program field [21:13]
<b>reqnum*</b>	Defines the number of the request line (0,1,...,7) to trigger either the HTU or the DMA. Default: 0 Location: Program field [25:23]
<b>request*</b>	Allows to select between no request (NOREQ), request (GENREQ) and quiet request (QUIET). See <a href="#">Section 19.2.9</a> . Default: No request Location: Control Field [28:27]

Request	C[28]	IC[27]	To HTU	To DMA
NOREQ	0	0	no request	no request
	1	0		
GENREQ	0	1	request	request
QUIET	1	1	quiet request	no request

<b>remote*</b>	Determines the 9-bit address of the remote address for the instruction. Default: Current instruction + 1 Location: Program field [8:0]
<b>control</b>	Determines whether the immediate data field [31:0] is cleared when it is read. When the bit is not set, reads do not clear the immediate data field. Default: OFF Location: Control field [26]
<b>en_pin_action*</b>	Determines whether the selected pin is ON so that the action occurs on the chosen pin Default: OFF Location: Control field [22]
<b>Cond_addr*</b>	Conditional address (optional): Defines the address of the next instruction when the condition occurs. Default: Current address + 1 Location: Control field [21:13]
<b>Pin*</b>	Pin Select: Selects the pin on which the action occurs. Enter the pin number. Default: pin 0 Location: Control field [12:8] except PCNT

The format CC{pin number} is also supported.

MSB				LSB	Description
0	0	0	0	0	Select HET[0]
0	0	0	0	1	Select HET[1]
(Each pin may be selected by writing its number in binary)					
1	1	1	1	0	Select HET[30]
1	1	1	1	1	Select HET[31]

**Reg\***

Register select: Selects the register for data comparison and storage

Default: No register (None)

Location: Control field [2:1] except for CNT instruction.

Extended Register Select C[7] is available for ACMP, ADC, ADD, ADM32, AND, DADM64, ECMP, ECNT, MCMP, MOV32, MOV64, OR, RADM64, SBB, SHFT, SUB, WCAP, WCAPE instructions.

Register	Ext Reg. C[7]	C[2]	C[1]
A	0	0	0
B	0	0	1
T	0	1	0
None	0	1	1
R	1	0	0
S	1	0	1
Reserved	1	1	0
Reserved	1	1	1

**Action\***

(2 Action Option) Either sets or clears the pin

Default: Clear

Location: Control Field [4]

Action	C[4]
Clear	0
Set	1

**Action\***

(4 Action Option) Either sets, clears, pulse high or pulse low on the pin. Set/clear are single pin actions, pulse high/low include the opposite pin action.

Default: Clear

Location: Control Field [4:3]

Action	Action Type	C[4]	C[3]
Clear	Set low on match	0	0
Set	Set high on match	1	0
Pulse Low	Set low on match + reset to high on Z=1 (opposite action)	0	1
Pulse High	Set high on match + reset to low on Z=1 (opposite action)	1	1

**hr\_lr\*** Specifies HIGH/LOW data resolution. If the hr\_lr field is HIGH, the instruction uses the hr\_data field. If the hr\_lr field is LOW, the hr\_data field is ignored.  
 Default: HIGH  
 Location: Program Field [8]

hr_lr	Prog. field [8]
LOW	1
HIGH	0

**prv\*** Specifies the initial value defining the previous bit (see [Section 19.2.5.8](#)). A value of ON sets the previous pin-level bit to 1. A value of OFF sets the initial value of the previous (prv) bit to 0. The prv bit is overwritten (set or reset) by the N2HET the first time the instruction is executed.

Default: OFF

Location: Control Field [25]

**cntl\_val\*** Available for DADM64, MOV64, and RADM64, this bit field allows the user to specify the replacement value for the remote control field.

**comp\_mode\*** Specifies the compare mode. This field is used with the 64-bit move instructions. This field ensures that the sub-opcodes are moved correctly.

Default: ECMP

Location: Control Field [6:5]

Action	C[6]	C[5]	order
ECMP	0	0	
SCMP	0	1	
MCMP1	1	0	REG_GE_DATA
MCMP2	1	1	DATA_GE_REG

### 19.5.3 Instruction Description

The following sections provide information for individual instructions.

Parameters in [] are optional. Please refer to the N2HET assembler user guide for the default values when parameters are omitted.

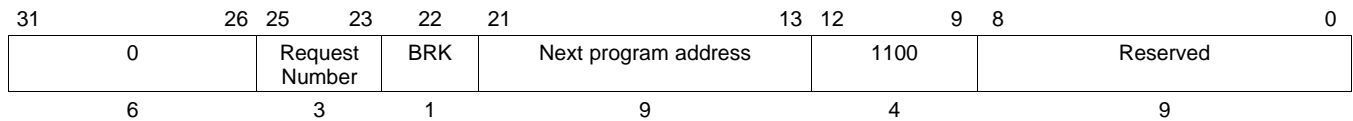
#### 19.5.3.1 ACMP (Angle Compare)

**Syntax**

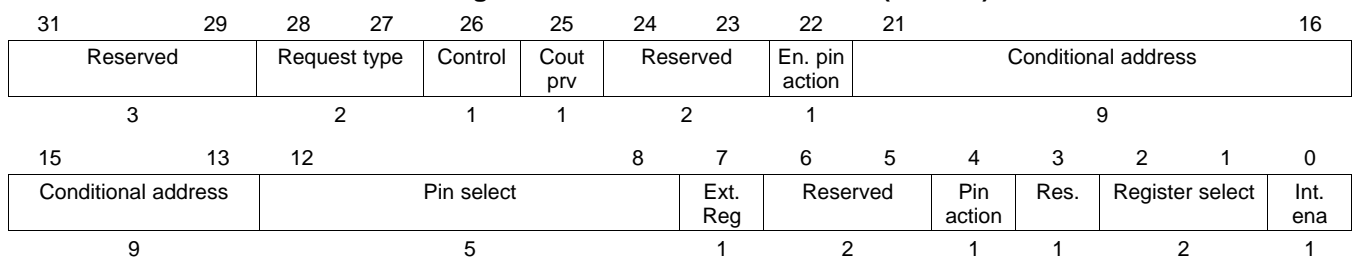
```

ACMP {
  [brk={OFF | ON}]
  [next={label | 9-bit unsigned integer}]
  [reqnum={3-bit unsigned integer}]
  [request={NOREQ | GENREQ | QUIET}]
  [control={OFF | ON}]
  [en_pin_action={OFF | ON}]
  [cond_addr={label | 9-bit unsigned integer}]
  pin={pin number}
  [action={CLEAR | SET}]
  reg={A | B | R | S | T | NONE}
  [irq ={OFF | ON}]
  data={25-bit unsigned integer}
}
    
```

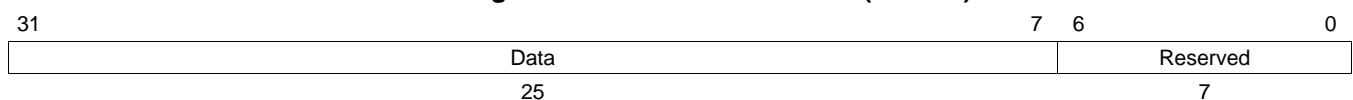
**Figure 19-68. ACMP Program Field (P31:P0)**



**Figure 19-69. ACMP Control Field (C31:C0)**



**Figure 19-70. ACMP Data Field (D31:D0)**



**Cycles** One

**Register modified** Selected register (A, B, R, S, or T)



The purpose of the comparison is to assert pin action when the angle compare value lies between the old counter value and the new counter value (held in the selected register). Since the angle increment varies from one loop resolution clock to another, an exact equality test cannot be applied. Instead, the following inequality is used to determine the occurrence of a match:

$$\text{Old counter value} < \text{Angle compare value} \leq \text{New counter value}$$

This is done by performing following comparisons:

Selected register value minus angle increment < angle compare value

Angle compare value ≤ Selected register value

<b>register</b>	Register B is recommended for typical applications with ACMP.
<b>irq</b>	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.
<b>data</b>	Specifies the 25-bit angle compare value.

### Execution

```

X = 0;
If (Data <= Selected Register)
    Cout = 0;
else
    Cout = 1;
If (Z == 0 AND (Selected Register - Angle Inc. < Data ) AND Cout == 0) OR
    (Z == 1 AND (Cout_prv == 1 OR Cout == 0))
{
    X = 1;
    If (Enable Pin Action == 1)
        Selected Pin = Pin Action AT next loop resolution clock;

    If (Interrupt Enable == 1)
        SW interrupt flag = 1;
    If ([C28:C27] == 01)
        Generate request on request line [P25:P23];
    If ([C28:C27] == 11)
        Generate quiet request on request line [P25:P23];

    Jump to Conditional Address;
}
else
    Jump to Next Program Address;

Cout_prv = Cout (always executed)

```

---

#### NOTE: Carry-Out Signal (Cout)

Cout is the carry-out signal of the adder. Even if it is not a flag, it is valid all along ACMP instruction execution.

---

Angle inc. = NAF\_global or hardware angle generator 11-bit input.

### 19.5.3.2 ACNT (Angle Count)

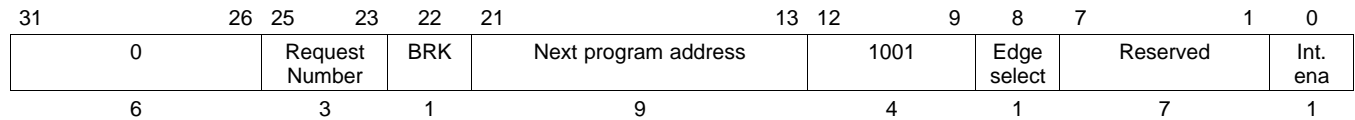
**Syntax**

```

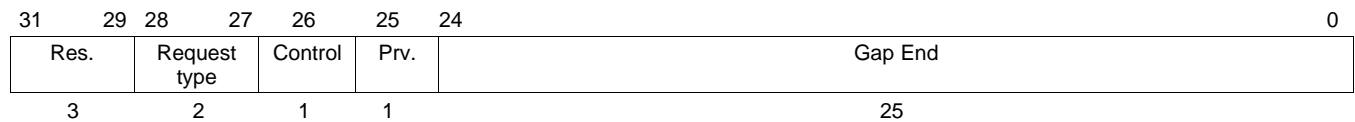
ACNT {
  [brk={OFF | ON}]
  [next={label | 9-bit unsigned integer}]
  [reqnum={3-bit unsigned integer}]
  [request={NOREQ | GENREQ | QUIET}]
  edge={RISING | FALLING}
  [irq = {OFF | ON}]
  [control={OFF | ON}]
  [prv={OFF | ON}]
  gapend = {25-bit unsigned integer}
  data={25-bit unsigned integer}
}

```

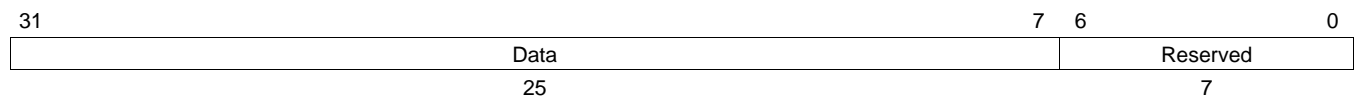
**Figure 19-71. ACNT Program Field (P31:P0)**



**Figure 19-72. ACNT Control Field (C31:C0)**



**Figure 19-73. ACNT Data Field (D31:D0)**



**Cycles** Two, as follows:

- First cycle: Angle increment condition and gap end comparison.
- Second cycle: Gap start comparison.

**Register modified** Register B (angle value)

**Description** This instruction defines a specialized virtual timer used after SCNT and APCNT to generate an angle-referenced time base that is synchronized to an external signal (that is, a toothed wheel signal). ACNT uses pin HET[2] exclusively. The edge select must be the same as the HET[2] edge which was selected in the previous APCNT.

ACNT refers to the same step width selection that the previous SCNT saved in flags SWF0 and SWF1 (see information on SCNT).

ACNT detects period variations of the external signal measured by APCNT and compensates related count errors.

A period increase is flagged in the deceleration flag (DCF). A period decrease is flagged in the acceleration flag (ACF). If no variation is detected, ACNT increments the counter value each time SCNT reaches its target.

If acceleration is detected, ACNT increments the counter value on each timer resolution. If deceleration is detected ACNT does not increment and is thus saturated.

ACNT also specifies the gap end angle value defining the end value of a gap range in ACNT where period measurements in APCNT are temporarily stopped to mask singularities in the external signal. ACNT uses register A containing gap start and register B to store the counter value.

**Edge** Specifies the edge for the input capture pin (HET[2]).

Action	P8	Edge Select
Rising	1	Detects a rising edge of HET[2]
Falling	0	Detects a falling edge of HET[2]

**irq** ON generates an interrupt when the edge state is satisfied and the gap flag is set. OFF prevents an interrupt from being generated.  
Default: OFF.

**gapend** Defines the 25-bit end value of a gap range. The start value is defined in the SCNT instruction.  
 $GAPEND = (\text{Step Value} * (\# \text{ of teeth on the toothed wheel} + \# \text{ of missing teeth})) - 1$

**data** Specifies the 25-bit initial count value for the data field.  
Default: 0.

---

#### NOTE: Target Edge Field

The target edge field represents the three LSBs of data field register in case of step width = 8, four LSBs for step width = 16, five LSBs for step width = 32 and six LSBs for step width = 64.

---

#### Execution

**Increment Condition:** ((Z = 1 AND DCF = 0) OR ACF = 1)

**Pin Edge Condition:** Specified edge detected on HET[2]

**Target Edge Condition:** (Target Edge field in data field = 0) AND (Angle Increment condition is true) AND (GPF = 0)

If (Angle Increment Condition) is false

```
{
    NAF = 0;
    Register B = Data field register;
}
else
{
    NAF = 1;
    If (Counter value != GapEnd)
    {
        Register B = Data field register + 1;
        Data Field Register = Counter value + 1;
    }
}
```

```

        else
        {
            Register B = 0;
            Data Field Register = 0;
            If (ACF == 0) DCF = 1;
        }
    }

Z = 0;

If (Data field register == GapStart)
{
    GPF = 1;
    If (Target Edge condition is true)
    {
        ACF = 0;
        If ((specified edge is not detected on pin HET[2]) AND (data
        field register != 0) AND (ACF == 0) AND (angle increment condition
        is true))
            DCF = 1;
    }
    If (specified edge is detected on pin HET[2])
    {
        DCF = 0;
        If ((target_edge_field != 0) AND (DCF == 0)) ACF = 1;
        If (GPF == 1)
        {
            GPF = 0;
            Z = 1;
            If (Interrupt Enable == 1)
                SW interrupt flag = 1;
            If ([C28:C27] == 01)
                Generate request on request line [P25:P23];;
            If ([C28:C27] == 11)
                Generate quiet request on request line
                [P25:P23];
        }
    }
}

If ((target_edge_field != 0) and (pin_edge_cond == 1))
{
    pin_update = 0;
}
else if (target_edge_field == 0)
{
    pin_update = 1;
}

If (pin_update is true in next loop clock cycle)
{
    Prv bit = Current Lx value of HET[2] pin;
}

Jump to next program address;

```

### 19.5.3.3 ADCNST (Add Constant)

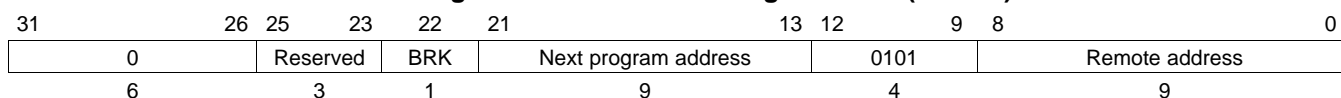
**Syntax**

```

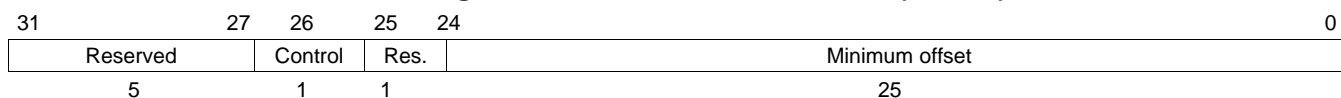
ADCNST {
  [brk={OFF | ON}]
  [next={label | 9-bit unsigned integer}]
  [control={OFF | ON}]
  remote={label | 9-bit unsigned integer}
  min_off={25-bit unsigned integer}
  data={25-bit unsigned integer}
  [hr_data={7-bit unsigned integer}]
}

```

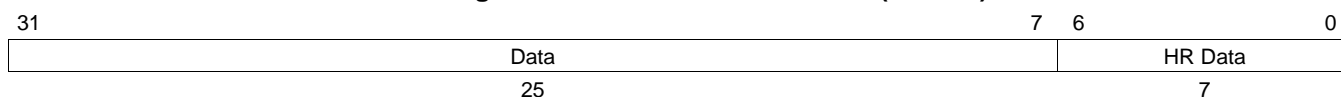
**Figure 19-74. ADCNST Program Field (P31:P0)**



**Figure 19-75. ADCNST Control Field (C31:C0)**



**Figure 19-76. ADCNST Data Field (D31:D0)**



**Cycles** Two

**Register modified** Register T (implicity)

**Description** ADCNST is an extension of ADM32. ADCNST first checks whether the data field value at the remote address is zero; it then performs different adds and moves on the result. ADCNST is typically used to extend the counter value of PWCNT.

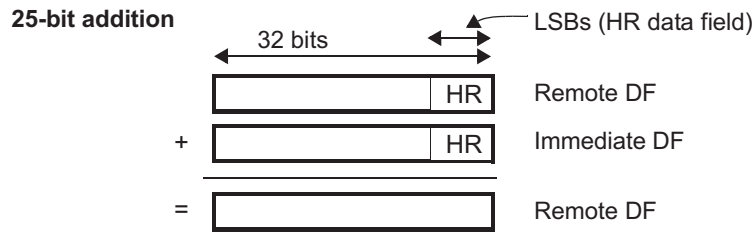
**min\_off** A 25-bit constant value that is added to the data field value if the remote data field is null.

**data** A 25-bit value that is always added to the remote data field.  
Default: 0.

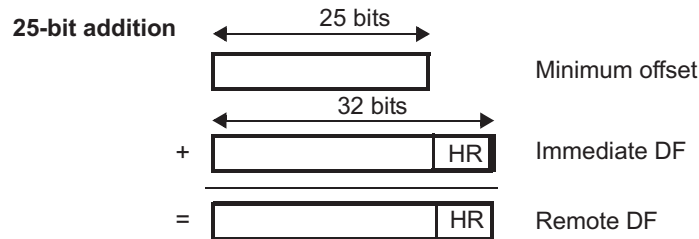
**hr\_data** Seven least significant bits of the data addition to the remote data field.  
Default: 0.

Figure 19-77 and Figure 19-78 illustrate the behavior of ADCNST if the remote data field is or is not zero.

**Figure 19-77. ADCNST Operation If Remote Data Field[31:7] Is Not Zero**



**Figure 19-78. ADCNST Operation if Remote Data Field [31:7] Is Zero**



**Execution**

```

If (Remote Data Field Value [31:7] != 0)
    Remote Data Field = Immediate Data Field + Remote Data Field;
else
    Remote Data Field = Immediate Data Field + min. offset(bits C24:C0);

Jump to Next Program Address;
    
```

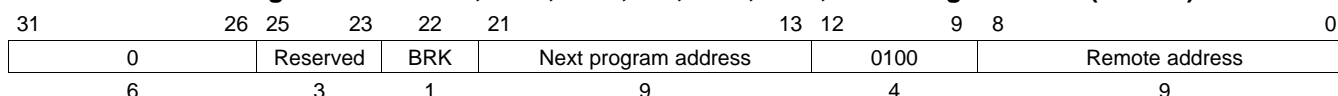
### 19.5.3.4 ADC, ADD, AND, OR, SBB, SUB, XOR

**Syntax**

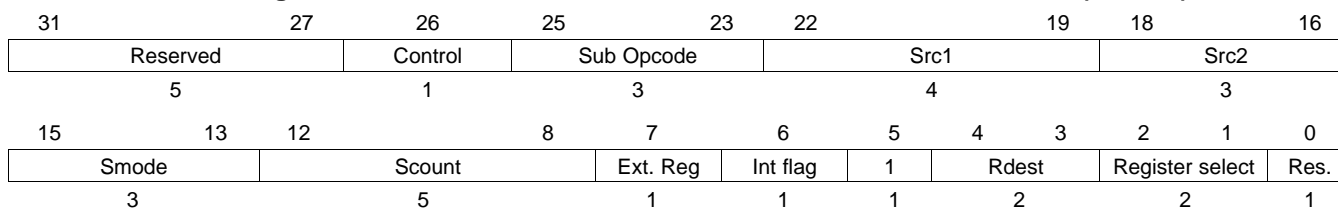
```

ADC | ADD | AND | OR | SBB | SUB | XOR {
src1 = { ZERO | IMM | A | B | R | S | T | ONES | REM | REMP }
src2 = { ZERO | IMM | A | B | R | S | T | ONES }
dest = { NONE | IMM | A | B | R | S | T }
[rdest = { NONE | REM | REMP }]
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
[remote={label | 9-bit unsigned integer}]
[control={OFF | ON}]
[init={OFF | ON}]
[smode = {LSL | CSL | LSR | CSR | RR | CRR | ASR }]
[scount = {5 bit unsigned integer}]
[data={25-bit unsigned integer}]
[hr_data={7-bit unsigned integer}]
}
    
```

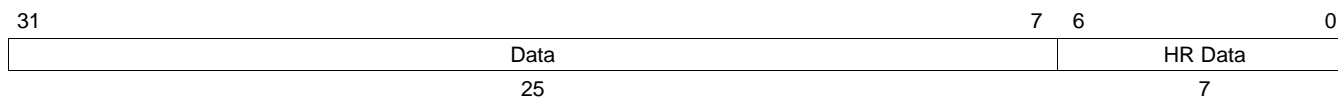
**Figure 19-79. ADC, ADD, AND, OR, SBB, SUB, XOR Program Field (P31:P0)**



**Figure 19-80. ADC, ADD, AND, OR, SBB, SUB, XOR Control Field (C31:C0)**



**Figure 19-81. ADC, ADD, AND, OR, SBB, SUB, XOR Data Field (D31:D0)**



**Cycles** One to three cycles, depending on operands selected. (See [Table 19-55](#))

**Register modified** Selected register (A, B, R, S, T, or NONE)

**Description** This instruction performs the specified 32-bit arithmetic or logical operation on operands src1 and src2, followed by an optional shift/rotate step. The result of this operation is then stored to either an N2HET register or the immediate data field of the instruction. In addition, the same result may be stored in a remote data field or the least significant bits of a remote instruction program field (P[8:0]). Bits P[8:0] of the program field are used by most instructions formats to hold the remote address that the instruction operates on, so the ability to update this field programatically makes it easier to write subroutines that operate on different data sets.

The Sub-Opcode field C[25:3] determines which type of operation (ADD, ADC, AND, OR, SBB, SUB, XOR) is executed by the instruction. A list of these operations and the corresponding Sub-Opcode encoding can be found in [Table 19-52](#).

All arithmetic is performed using 32-bit integer math. However, source and destination operands vary in width and can be 9 bits (REMP), 25 bits (A, B) or 32 bits (R, S, T, IMM, REM). Source operands REMP, A, B are extended to 32-bits before being operated on. Also the result of the computation needs to be truncated before being written back to REMP, A, or B when these are selected as destination operands. [Table 19-53](#) provides a list of source operand options, how they are expanded to 32-bit integers (if applicable) and the control field encoding to select the option for src1 and src2 operands.

[Table 19-54](#) provides a similar list of destination operands and their encodings. Up to two destination operands may be selected for each instruction, a register/immediate destination and a remote destination may be selected simultaneously. Truncation is performed independently for each destination operand as appropriate to its size.

An optional shift step following the arithmetic or logical operation may be selected through the smode and scout operands. The shift or rotate type is selected by the smode field; [Figure 19-82](#) illustrates the options that are available for smode. The number of bits shifted is determined by the scout operand.

**Table 19-52. Arithmetic / Bitwise Logic Sub-Opcodes**

Instruction	Description	Operation	Sub-Opcode
ADC	Add with Carry	result = src1 + src2 + C	C[25:23] = 011
ADD	Add	result = src1 + src2	C[25:23] = 001
AND	Bitwise Logic And	result = src1 & src2	C[25:23] = 010
OR	Bitwise Logic Or	result = src1   src2	C[25:23] = 100
SBB	Subtract with Borrow	result = src1 - src2 - C	C[25:23] = 110
SUB	Subtract	result = src1 - src2	C[25:23] = 101
XOR	Bitwise Logic Exclusive Or	result = src1 ^ src2	C[25:23] = 111

**Table 19-53. Source Operand Choices**

Source Operand	32-bit value	Address	src1	src2
A	{A[24:0], 0x00}	n/a	C[22:19] = 0010	C[18:16] = 010
B	{B[24:0], 0x00}	n/a	C[22:19] = 0011	C[18:16] = 011
R	R[31:0]	n/a	C[22:19] = 0100	C[18:16] = 100
S	S[31:0]	n/a	C[22:19] = 0101	C[18:16] = 101
T	T[31:0]	n/a	C[22:19] = 0110	C[18:16] = 110
IMM	D[31:0]	current instruction address	C[22:19] = 0001	C[18:16] = 001
ZERO	0x00000000	n/a	C[22:19] = 0000	C[18:16] = 000
ONES	0xFFFFFFFF	n/a	C[22:19] = 0111	C[18:16] = 111
REM	D[31:0]	specified by remote[8:0]	C[22:19] = 1000	n/a
REMP	{0x000000, P[8:0]}	specified by remote[8:0]	C[22:19] = 1001	n/a

**Table 19-54. Destination Operand Choices**

Destination Operand	Stored Value	Address	dest	rdest
A	A[24:0] = result [31:8]	n/a	C[7] = 0, C[2:1] = 00	n/a
B	B[24:0] = result [31:8]	n/a	C[7] = 0, C[2:1] = 01	n/a
R	R[24:0] = result [31:0]	n/a	C[7] = 1, C[2:1] = 00	n/a
S	S[24:0] = result [31:0]	n/a	C[7] = 1, C[2:1] = 01	n/a
T	T[24:0] = result [31:0]	n/a	C[7] = 0, C[2:1] = 10	n/a



**Table 19-54. Destination Operand Choices (continued)**

Destination Operand	Stored Value	Address	dest	rdest
IMM	D[31:0] = result [31:0]	current instruction address	C[7] = 1, C[2:1] = 10	n/a
NONE	n/a	n/a	C[7] = 0, C[2:1] = 11	C[4:3] = 00
REM	D[31:0] = result [31:0]	specified by remote[8:0]	n/a	C[4:3] = 01
REMP	P[8:0] = result [8:0]	specified by remote[8:0]	n/a	C[4:3] = 10

**Figure 19-82. Shift Encoding**

Shift Type	C[15:13] smode	Operation Illustrated <sup>a</sup>
No Shift Applied	0 0 0	n/a - no shift
ASR - Arithmetic Shift Right	001	
LSL - Logical Shift Left	010	
CSL - Carry Shift Left	011	
LSR - Logical Shift Right	100	
CSR - Carry Shift Right	01	
RR - Rotate Right	110	
CRR - Carry Rotate Right	11	

a. IC1 is the carry flag after the arithmetic / logical operation is performed. IC2 is the updated carry flag after the shift operation is performed. s is the sign bit.

**Table 19-55. Execution Time for ADC, ADD, AND, OR, SBB, SUB, XOR Instructions**

src1	dest	rdest	remote[8:0]	Cycle s
ZERO, IMM, A, B, R, S, T, or ONES	A,B,R,S,T, or NONE	NONE	!= next[8:0]	1
REM or REMP	A,B,R,S,T, or NONE	NONE	!= next[8:0]	2
ZERO, IMM, A, B, R, S, T, or ONES	IMM	REM	!= next[8:0]	2
ZERO, IMM, A, B, R, S, T, or ONES	A,B,R,S,T, or NONE	REMP	!= next[8:0]	2
ZERO, IMM, A, B, R, S, T, or ONES	A,B,R,S,T, or NONE	NONE	== next[8:0]	2
REM or REMP	IMM	REM	x	3
x	IMM	REMP	x	3
REM or REMP	x	REM	== next[8:0]	3
x	x	REMP	== next[8:0]	3

**Execution**

```

        / Notes: IR1, IR2 are 32-bit intermediate results
// SRC1, SRC2 are 32-bit sources selected
//          by fields src1, src2
// IC1, IC2 are intermediate values of the carry flag
// IZ1, IZ2 are intermediate values of the zero flag
// IN1, IN2 are intermediate values of the negative flag
// IV1, IV2 are intermediate values of the overflow flag
// scout is the shift count (0 to 31) specified by C12:C8

/***** SOURCE OPERAND DECODING STAGE *****/
switch (C22:C19)
{
    case 0000:SRC1[31:0] = 0x00000000
    case 0001:SRC1[31:0] = Immediate Data Field D[31:0]
    case 0010:SRC1[31:8] = A[24:0]; SRC1[6:0] = 0
    case 0011:SRC1[31:8] = B[24:0]; SRC1[6:0] = 0
    case 0100:SRC1[31:0] = R[31:0]
    case 0101:SRC1[31:0] = S[31:0]
    case 0110:SRC1[31:0] = T[31:0]
    case 0111:SRC1[31:0] = 0xFFFFFFFF
    case 1000:SRC1[31:0] = Remote Data Field D[31:0]
    case 1001:SRC1[31:9] = 0; SRC1[8:0] = Remote Program Field P[8:0]
}

switch (C18:C16)
{
    case 000:SRC2[31:0] = 0x00000000
    case 001:SRC2[31:0] = Immediate Data Field[31:0]
    case 010:SRC2[31:8] = A[24:0]; SRC2[6:0] = 0
    case 011:SRC2[31:8] = B[24:0]; SRC2[6:0] = 0
    case 100:SRC2[31:0] = R[31:0]
    case 101:SRC2[31:0] = S[31:0]
    case 110:SRC2[31:0] = T[31:0]
    case 111:SRC2[31:0] = 0xFFFFFFFF
}

/***** ARITHMETIC / LOGICAL OPERATION STAGE *****/
switch (C[25:23])
{
    case 011:IR1 = src1 + src2 + C // ADC
    case 001:IR1 = src1 + src2 // ADD
    case 010:IR1 = src1 & src2 // AND
    case 100:IR1 = src1 | src2 // OR
    case 110:IR1 = src1 - src2 - C // SBB
    case 101:IR1 = src1 - src2 // SUB
    case 111:IR1 = src1 ^ src2 // XOR
}

IC1 = Carry Out if Operation is ADD, ADC, SUB, SBB
      = 0 if Operation is AND, OR, XOR
IZ1 = Set if IR1 is zero, Clear if IR1 is non-zero
IN1 = IR[31]
IV1 = (IC1 XOR IR1[31]) AND NOT(SRC1[31] XOR SRC2[31])

/***** SHIFT STAGE *****/
switch (C15:C13)
{
    case 000: // smode = No Shift
        IR2 = IR1
        IC2 = IC1; IZ2 = IZ1; IN2 = IN1; IV2 = IV1;

    case 001: // smode = Arithmetic Shift Right
        IR2[31 - scout : 0] = IR1[31:scout]

        if (scout>0) {

```

```

        IR2[31 : 31 - scount + 1] = IR1[31]
        IC2 = IR1[scount-1]
    }
    else {
        IC2 = IC1
    }

    IN2 = IR2[31];
    if (IR2 == 0) { IZ2 = 1 } else {IZ2 = 0};
    IV2 = (IR2[31] XOR IR1[31]) OR IV1

case 010: // smode = Logical Shift Left
    IR2[31 : scount] = IR1[31 - scount: 0]

    if (scount > 0) {
        IR2[scount - 1 : 0] = 0
    }

    IC2 = IC1
    IN2 = IR2[31];
    if (IR2 == 0) { IZ2 = 1 } else {IZ2 = 0};
    IV2 = (IR2[31] XOR IR1[31]) OR IV1

case 011: // smode = Carry Shift Left
    IR2[31 : scount] = IR1[31 - scount: 0]

    if (scount>0) {
        IR2[scount - 1 : 0] = [IC1,...IC1]
        IC2 = IR1[31 - scount + 1]
    }
    else
    {
        IC2 = IC1
    }

    IN2 = IR2[31];
    if (IR2 == 0) { IZ2 = 1 } else {IZ2 = 0};
    IV2 = (IR2[31] XOR IR1[31]) OR IV1

case 100: // smode = Logical Shift Right
    IR2[31 - scount : 0] = IR1[31:scount]

    if (scount>0) {
        IR2[31 : 31 - scount + 1] = 0
    }

    IC2 = IC1
    IN2 = IR2[31];
    if (IR2 == 0) { IZ2 = 1 } else {IZ2 = 0};
    IV2 = (IR2[31] XOR IR1[31]) OR IV1

case 101: // smode = Carry Shift Right
    IR2[31 - scount : 0] = IR1[31:scount]

    if(scount>0) {
        IR2[31:31-scount + 1] = [IC1,...IC1]
        IC2 = IR1[scount-1]
    }
    else {
        IC2 = IC1
    }

    IN2 = IR2[31];
    IZ2 = Set if IR2 == 0;
    IV2 = (IR2[31] XOR IR1[31]) OR IV1

```

```

case 110: // smode = Rotate Right
    IR2[31 - scount : 0] = IR1[31:scount]

    if(scount>0) {
        IR2[31:31-scount+1] = IR1[scount-1:0]
        IC2 = IR1[scount-1]
    }
    else {
        IC2 = IC1
    }

    IN2 = IR2[31];
    if (IR2 == 0) { IZ2 = 1 } else {IZ2 = 0};
    IV2 = (IR2[31] XOR IR1[31]) OR IV1

case 111: // smode = Carry Rotate Right
    IR2[31 - scount : 0] = IR1[31:scount]

    if (scount == 0) {
        IC2 = IC1
    }
    else if (scount == 1) {
        IR2[31] = IC1
        IC2 = IR1[0]
    }
    else {
        IR2[31:31-scount+1] = {IR1[scount-2:0],IC1}
        IC2 = IR1[scount - 1]
    }

    IN2 = IR2[31];
    if (IR2 == 0) { IZ2 = 1 } else {IZ2 = 0};
    IV2 = (IR2[31] XOR IR1[31]) OR IV1
}

/***** WRITE REGISTER DESTINATION STAGE *****/
switch (C7, C2:C1)
{
    case 000:A[24:0] = IR2[31:8]
    case 001:B[24:0] = IR2[31:8]
    case 010:T[31:0] = IR2[31:0]
    case 011:IR2 is not stored in register, immediate
    case 100:R[31:0] = IR2[31:0]
    case 101:S[31:0] = IR2[31:0]
    case 110:Immediate Data Field[31:0] = IR2
    case 111:IR2 is not stored in register, immediate
}
/***** WRITE REMOTE DESTINATION STAGE *****/
switch (C4:3)
{
    case 00:IR2 is not stored in remote field
    case 01:Remote Data Field D[31:0] = IR2
    case 10:Remote Program Field P[8:0] = IR2[8:0]
    case 11:IR2 is not stored in remote field
}
/***** UPDATE FLAGS STAGE *****/
C FLAG = IC2
N FLAG = IN2
Z FLAG = IZ2
V FLAG = IV2
If (Init Flag == 1)
{
    ACF = 0;
    DCF = 1;
    GPF = 0;
    NAF = 0;
}

```

```
}  
else ACF, DCF, GPF, NAF remain unchanged;
```

19.5.3.5 ADM32 (Add Move 32)

**Syntax**

```

ADM32 {
  [brk={OFF | ON}]
  [next={label | 9-bit unsigned integer}]
  remote={label | 9-bit unsigned integer}
  [control={OFF | ON}]
  [init={OFF | ON}]
  type={IM&REGTOREG | REM&REGTOREG | IM&REMTOREG |
  IM&REGTOREM}
  reg={A | B | R | S | T | NONE}
  data={25-bit unsigned integer}
  [hr_data={7-bit unsigned integer}]
}
    
```

Figure 19-83. ADM32 Program Field (P31:P0)

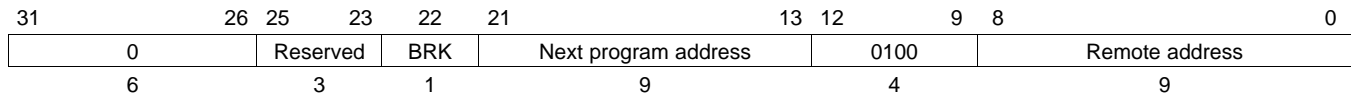


Figure 19-84. ADM32 Control Field (C31:C0)

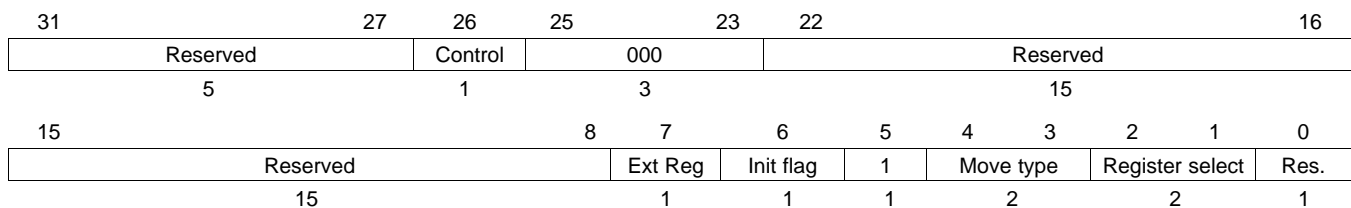
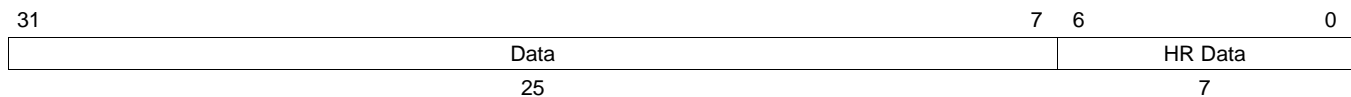


Figure 19-85. ADM32 Data Field (D31:D0)



**Cycles** One or two cycles (see [Table 19-56](#))

**Register modified** Selected register (A, B, R, S, or T)

**Description** This instruction modifies the selected ALU register or data field values at the remote address depending on the move type. The modified value results from adding the immediate or remote data field to the ALU register or the remote data field, depending on the move type. Table description shows the C2 and C1 bit encoding for determining which register is selected.

**init** (Optional) Determines whether or not system flags are initialized. A value of ON reinitializes the following system flags to these states:  
 Acceleration flag (ACF) = 0  
 Deceleration flag (DCF) = 1

Gap flag (GPF) = 0  
 New angle flag (NAF) = 0  
 A value of OFF results in no change to the system flags.  
 Default: OFF

**type** Specifies the move type to be executed.

**Table 19-56. Move Types for ADM32**

Type	C4	C3	Add	Destination(s)	Cycles
IM&REGTOREG	0	0	Imm. data field + Reg. A, B, R, S, or T	Register A, B, R, S, or T	1
REM&REGTOREG	0	1	Remote data field + Reg. A, B, R, S, or T	Register A, B, R, S, or T	2
IM&REMTOREG	1	0	Imm. data field + Remote data field	Register A, B, R, S, or T	2
IM&REGTOREM	1	1	Imm. data field + Reg. A, B, R, S, or T	Remote data field	1

If selected register is R, S, or T, the operation is a 32-bit Addition/move. If A or B register is selected, it is limited to 25-bit operation since A and B only support 25-bit.

**data** Specifies the 25-bit integer value for the immediate data field.

**hr\_data** Specifies the 7 least significant bits of the immediate data field.  
 Default: 0.

**Execution**

```

switch (C4:C3)
{
  case 00:
    Selected register = Selected register + Immediate Data Field;
  case 01:
    Selected register = Selected register + Remote Data Field;
  case 10:
    Selected register = Immediate Data Field + Remote Data Field;
  case 11:
    Remote Data Field = Selected register + Immediate Data Field;
}

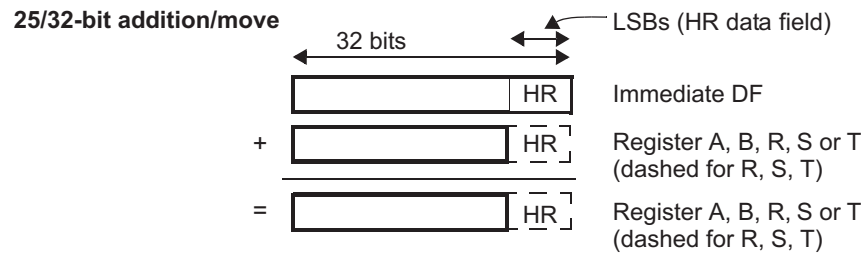
If (Init Flag == 1)
{
  ACF = 0;
  DCF = 1;
  GPF = 0;
  NAF = 0;
}
else
  All flags remain unchanged;

Jump to Next Program Address;

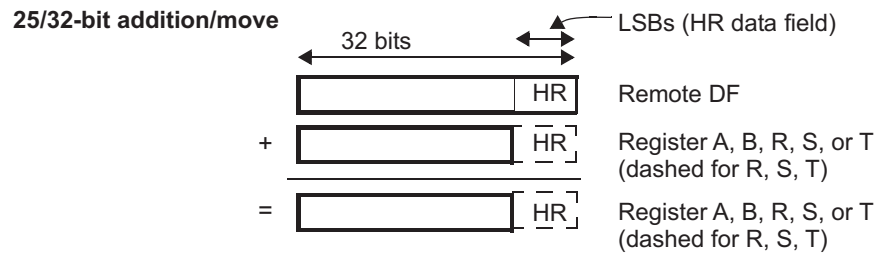
```

Figure 19-86 and Figure 19-87 illustrate the ADM32 operation for various cases.

**Figure 19-86. ADM32 Add and Move Operation for IM&REGTOREM (Case 00)**



**Figure 19-87. ADM32 Add and Move Operation for REM&REGTOREG (Case 01)**





### 19.5.3.6 APCNT (Angle Period Count)

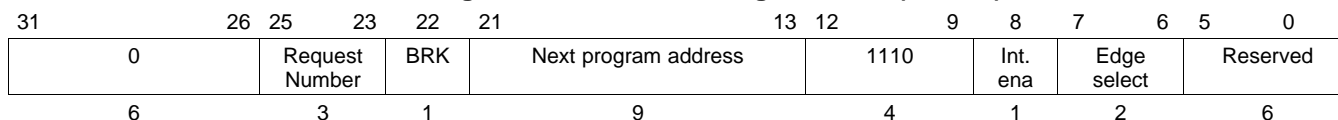
**Syntax**

```

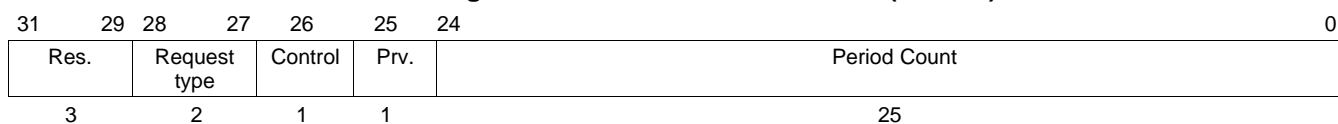
APCNT {
  [brk={OFF | ON}]
  [next={label | 9-bit unsigned integer}]
  [reqnum={3-bit unsigned integer}]
  [request={NOREQ | GENREQ | QUIET}]
  [irq={OFF | ON}]
  type={FALL2FALL | RISE2RISE}
  [control={OFF | ON}]
  prv={OFF | ON}}
  period={25-bit unsigned integer}
  data={25-bit unsigned integer}
}

```

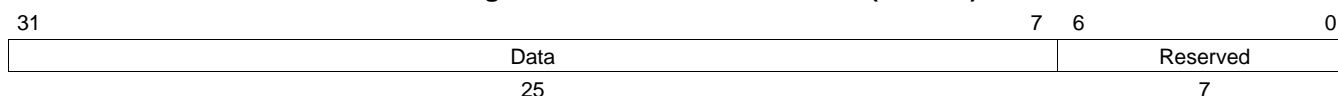
**Figure 19-88. APCNT Program Field (P31:P0)**



**Figure 19-89. APCNT Control Field (C31:C0)**



**Figure 19-90. APCNT Data Field (D31:D0)**



**Cycles** One or two cycles

- Cycle 1: edge detected (normal operation)
- Cycle 2: edge detected and GPF = 1 and underflow condition is true

One cycle (normal operation) two cycles (edge detected)

**Register modified** Register A and T (implicitly)

**Description** This instruction is used before SCNT and ACNT to generate an angle-referenced time base synchronized to an external signal (that is, a toothed wheel signal). It is assumed that the pin and edge selections are the same for APCNT and ACNT.

APCNT is restricted to pin HET[2]. The toothed wheel must then be connected to pin HET[2].

APCNT uses the gap flag (GPF) defined by ACNT to start or stop captures in the period count field [C24:C0]. When GPF = 1, the previous period value is held in the control field and in register T. When GPF = 0, the current period value is captured in the control field and in register T.

APCNT uses the step width flags (SWF0 and SWF1) defined by SCNT to detect period durations shorter than one step, and then disables capture. The edge select encoding is shown in [Table 19-57](#).

<b>irq</b>	ON generates an interrupt when the edge state is satisfied. OFF prevents an interrupt from being generated. Default: OFF.
<b>type</b>	Specifies the edge type that triggers the instruction. Default: Fall2Fall.

**Table 19-57. Edge Select Encoding for APCNT**

type	P7	P6	Selected Condition
Fall2Fall	1	0	Falling edge
Rise2Rise	1	1	Rising edge

<b>period</b>	Contains the 25-bit count value from the previous APCNT period.
<b>data</b>	25-bit value serving as a counter. Default: 0.

### Execution

```

Z = 0;

If (Data field register != 1FFFFFFh)
{
    Register A = Data field register + 1;
    Data field register = Data field register + 1;
}
elseif (specified edge not detected on HET[2])
{
    Register A = 1FFFFFFh;
    APCNT Ovflw flag = 1;
}

If (specified edge detected on HET[2])
{
    Z = 1;

    If (Data field register == 1FFFFFFh)
    {
        Register A = 1FFFFFFh;
        Register T = 1FFFFFFh; Period count = 1FFFFFFh;
        Period count = 1FFFFFFh;
    }
    elseif (GPF == 0 AND Data Field register >= Step width)
    {
        Register A = Data field register + 1;
        Register T = Register A;
        Period count = Register T;

        If (Interrupt Enable == 1)
            SW interrupt flag = 1;
        If ([C28:C27] == 01)
            Generate request on request line [P25:P23];
        If ([C28:C27] == 11)
            Generate quiet request on request line [P25:P23];
    }
}

```

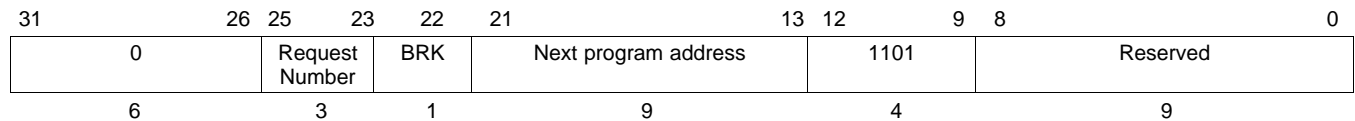
```
    }  
    If (GPF == 1)  
        Register T = Period count;  
    If (Data Field register < Step width)  
    {  
        Register T = Period count;  
        APCNT Undflw flag = 1;  
        Period Count = 000000h;  
    }  
  
    Data field register = 000000h;  
}  
else  
{  
    Register T = Period count;  
}  
  
Prv bit = Current Lx value of HET[2] pin;  
  
Jump to Next Program Address;
```

### 19.5.3.7 BR (Branch)

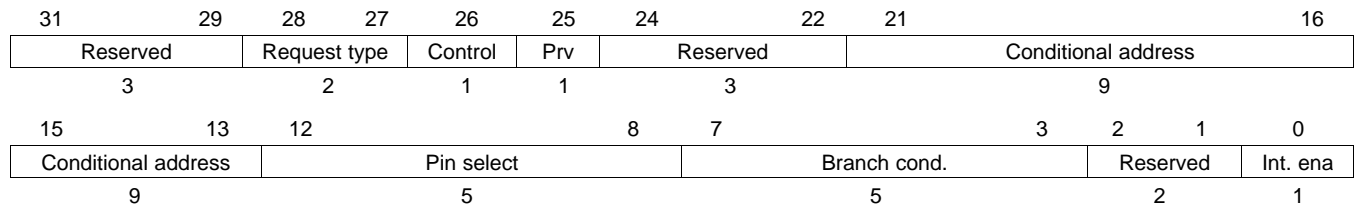
**Syntax**

```
BR {
  [brk={OFF | ON}]
  [next={label | 9-bit unsigned integer}]
  [reqnum={3-bit unsigned integer}]
  [request={NOREQ | GENREQ | QUIET}]
  [control={OFF | ON}]
  [prv={OFF | ON}]
  cond_addr={label | 9-bit unsigned integer}
  [pin= {pin number}]
  event={NOCOND | FALL | RISE | BOTH | ZERO | NAF | LOW | HIGH | C | NC
  | EQ | Z | NE | NZ | N | PZ | V | NV | ZN | P | GE | LT | GT | LE | LO | HS }
  [irq={OFF | ON}]
}
```

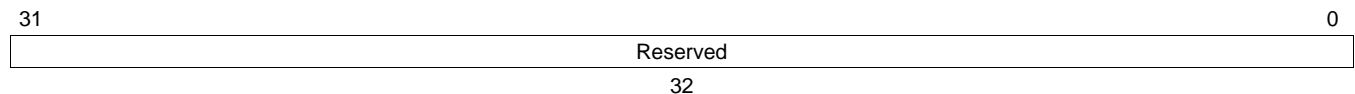
**Figure 19-91. BR Program Field (P31:P0)**



**Figure 19-92. BR Control Field (C31:C0)**



**Figure 19-93. BR Data Field (D31:D0)**



**Cycles**

One

**Register modified**

None

**Description**

This instruction executes a jump to the conditional address [C21:C13] on a pin or a flag condition, and can be used with all pins.

[Table 19-58](#) provides the branch condition encoding.

<b>event</b>	Specifies the event that triggers a jump to the indexed program address. Default: FALL
<b>irq</b>	ON generates an interrupt when the event occurs that triggers the jump. If irq is set to OFF, no interrupt is generated. Default: OFF.

**Table 19-58. Branch Condition Encoding for BR**

event	C7	C6	C5	C4	C3	Branch Condition
NOCOND	0	0	0	0	0	Always
FALL	0	0	1	0	0	On falling edge on the selected pin
RISE	0	1	0	0	0	On rising edge on selected pin
BOTH	0	1	1	0	0	On rising or falling edge on selected pin
ZERO	1	0	0	0	0	If Zero flag is set
NAF	1	0	1	0	0	If NAF_global flag is set
LOW	1	1	0	0	0	On LOW level on selected pin
HIGH	1	1	1	0	0	On HIGH level on selected pin
C	0	0	0	0	1	Carry Set: C==1
NC	0	0	0	1	1	Carry Not Set: C==0
EQ, Z	0	0	1	0	1	Equal or Zero: Z==1
NE, NZ	0	0	1	1	1	Not Equal or Not Zero: Z==0
N	0	1	0	0	1	Negative: N==1
PZ	0	0	1	1	1	Positive or Zero: N==0
V	0	1	1	0	1	Overflow: V==1
NV	0	1	1	1	1	No Overflow: V==0
ZN	1	0	0	0	1	Zero or Negative: (Z OR N) == 1
P	1	0	0	1	1	Positive: (Z OR N) == 0
GE	1	0	1	1	1	Signed Greater Than or Equal: (N XOR V) == 0
L	1	0	1	0	1	Signed Less Than (N XOR V) == 1
G	1	1	0	1	1	Signed Greater Than (Z OR (N XOR V)) == 0
LE	1	1	0	0	1	Signed Less Than (Z OR (N XOR V)) == 1
LO	1	1	1	1	1	Unsigned Less Than: (C OR Z) == 0
HS	1	1	1	0	1	Unsigned Higher or Same (C OR Z) == 1

**Execution**

```

If (Condition is true)
{
  If (Interrupt Enable == 1) SW interrupt flag = 1;
  If ([C28:C27] == 01) Generate request on request line [P25:P23];
  If ([C28:C27] == 11) Generate quiet request on request line [P25:P23];
  Jump to Conditional Address;
}
else
{
  Jump to Next Program Address;
}

Prv bit = Current Lx value of selected pin; (Always Executed)

```

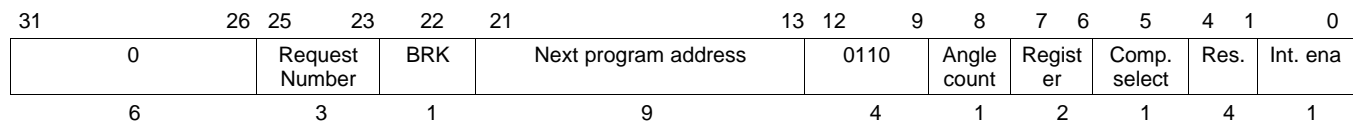
### 19.5.3.8 CNT (Count)

**Syntax**

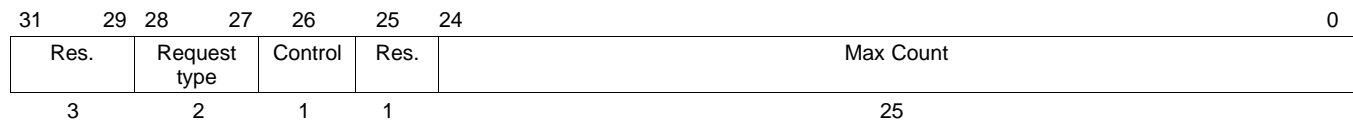
```

CNT {
  [brk={OFF | ON}]
  [next={label | 9-bit unsigned integer}]
  [reqnum={3-bit unsigned integer}]
  [request={NOREQ | GENREQ | QUIET}]
  [angle_count={OFF | ON}]
  [reg={A | B | T | NONE}]
  [comp ={EQ | GE}]
  [irq={OFF | ON}]
  [control={OFF | ON}]
  max={25-bit unsigned integer}
  [data={25-bit unsigned integer}]
}
    
```

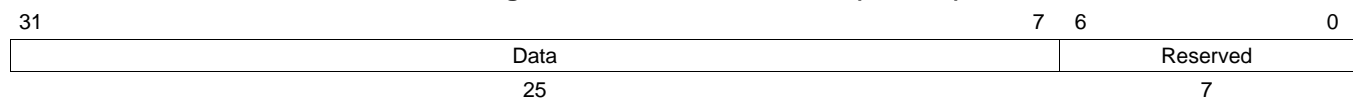
**Figure 19-94. CNT Program Field (P31:P0)**



**Figure 19-95. CNT Control Field (C31:C0)**



**Figure 19-96. CNT Data Field (D31:D0)**



**Cycles** One or two  
One cycle (time mode), two cycles (angle mode)

**Register modified** Selected register (A, B or T)

**Description** This instruction defines a virtual timer. The counter value stored in the data field [D31:7] is incremented unconditionally on each resolution when in time mode (angle count bit [P8] = 0). When the count reaches the maximum count specified in the control field, the counter is reset. It takes one cycle in this mode.

In angle mode (angle count bit [P8] = 1), CNT needs data from the software angle generator (SWAG). When in angle count mode the angle increment value will be 0 or 1. It takes two cycles in this mode.

<b>angle_count</b>	Specifies when the counter is incremented. A value of ON causes the counter value to be incremented only if the new angle flag is set (NAF_global = 1). A value of OFF increments the counter each time the CNT instruction is executed. Default value for this field is OFF.
<b>comp</b>	When set to EQ the counter is reset, when it is equal to the maximum count. When set to GE the counter is reset, when it is greater or equal to the maximum count. Default: GE.
<b>irq</b>	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.
<b>max</b>	Specifies the 25-bit integer value that defines the maximum count value allowed in the data field. When the count in the data field is equal to max, the data field is reset to 0 and the Z system flag is set to 1.
<b>data</b>	Specifies the 25-bit integer value serving as a counter. Default: 0.

### Execution

```

Z = 0;

If (Angle Count (bit P8 == 1))
{
  If (NAF_global == 0)
  {
    Selected register = immediate data field;
    Jump to Next Program Address;
  }
  else
  {
    If ((Immediate Data Field + Angle Increment) >= Max count)
    {
      Z = 1;
      Selected register = ((Immediate Data Field + Angle Inc.) - Max count);
      Immediate Data Field = ((Immediate Data Field + Angle Inc.) - Max count);

      If (Interrupt Enable == 1) SW interrupt flag = 1;
      If ([C28:C27] == 01) Generate request on request line [P25:P23];
      If ([C28:C27] == 11) Generate quiet request on request line [P25:P23];
    }
    else
    {
      Selected register = Immediate Data Field + Angle Increment;
      Immediate Data Field = Immediate Data Field + Angle Increment;
    }
  }
}

else if(Time mode (bit P8 == 0))
{
  If [(P5==0) AND (Immediate Data Field == Max count)]
  OR [(P5==1) AND (Immediate Data Field >= Max count)]
  {
    Z = 1;
  }
}

```

```
Selected register = 00000;
Immediate Data Field = 00000;

If (Interrupt Enable == 1) SW interrupt flag = 1;
If ([C28:C27] == 01) Generate request on request line [P25:P23];
If ([C28:C27] == 11) Generate quiet request on request line [P25:P23];
}
else
{
Selected register = Immediate Data Field + 1;
Immediate Data Field = Immediate Data Field + 1;
}
}

Jump to Next Program Address;
```



### 19.5.3.9 DADM64 (Data Add Move)

**Syntax**

```

DADM64 {
  [brk={OFF | ON}]
  [next={label | 9-bit unsigned integer}]
  remote={label | 9-bit unsigned integer}
  [request={NOREQ | GENREQ | QUIET}]
  [control={OFF | ON}]
  [en_pin_action={OFF | ON}]
  [cond_addr={label | 9-bit unsigned integer}]
  [pin={pin number}]
  comp_mode={ECMP | SCMP | MCMP1 | MCMP2}
  [action={CLEAR | SET | PULSELO | PULSEHI}]
  [reg={A | B | R | S | T | NONE}]
  [irq={OFF | ON}]
  [data={25-bit unsigned integer}]
  [hr_data= {7-bit unsigned integer}]
}
    
```

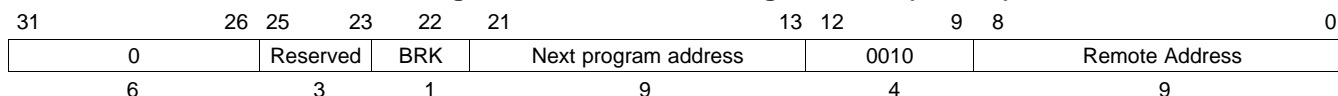
-or-

**Syntax**

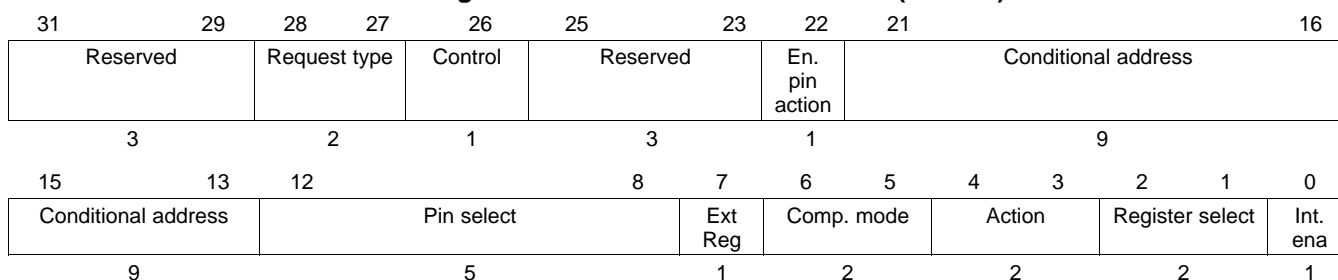
```

DADM64 {
  [brk={OFF | ON}]
  [next={label | 9-bit unsigned integer}]
  remote={label | 9-bit unsigned integer}
  cntl_val={29-bit unsigned integer}
  data={25-bit unsigned integer}
  [hr_data= {7-bit unsigned integer}]
}
    
```

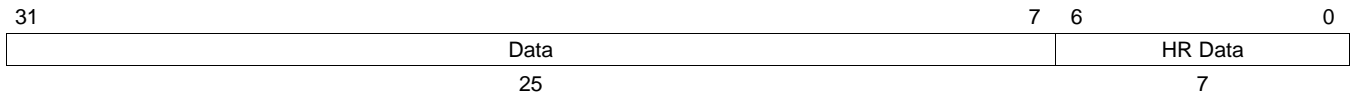
**Figure 19-97. DADM64 Program Field (P31:P0)**



**Figure 19-98. DADM64 Control Field (C31:C0)**



**Figure 19-99. DADM64 Data Field (D31:D0)**



**Cycles** Two

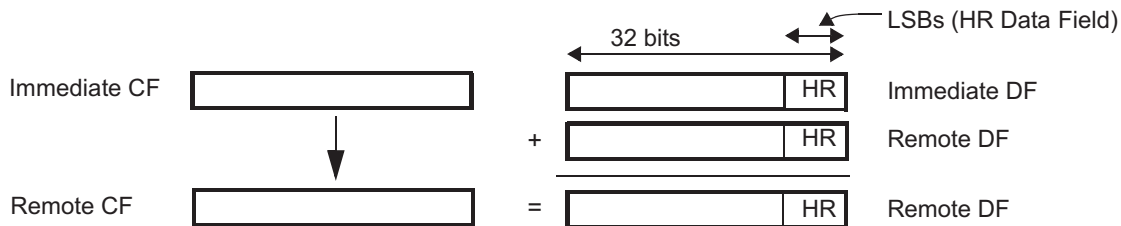
**Register modified** Register T (implicitly)

**Description** This instruction modifies the data field and the control field at the remote address. The remote data field value is not just replaced, but is added with the DADM64 data field.

DADM64 has two distinct syntaxes. In the first syntax, bit values may be set by assigning a value to each of the control fields. This syntax is convenient for modifying control fields that are arranged similarly to the format of the DADM64 control field. A second syntax, in which the entire 29-bit control field is specified by the `cntl_val` field, is convenient when the remote control field is dissimilar to the DADM64 control field. Either syntax may be used, but you must use one or the other but not a combination of syntaxes.

Figure 19-100 shows the DADM64 add and move operation.

**Figure 19-100. DADM64 Add and Move Operation**



**Table 19-59. DADM64 Control Field Description**

request	maintains the control field for the remote instruction
control	maintains the control field for the remote instruction
en_pin_action	maintains the control field for the remote instruction
cond_addr	maintains the control field for the remote instruction
pin	maintains the control field for the remote instruction
register	maintains the control field for the remote instruction
action	maintains the control field for the remote instruction
irq	maintains the control field for the remote instruction
data	Specifies the 25-bit initial value for the data field.
hr_data	Seven least significant bits of the 32 bit data field. Default: 0
cntl_val	Specifies the 29 least significant bits of the Control field.

**Execution**

```
Remote Data Field = Remote Data Field + Immediate Data Field;
Register T = Immediate Data Field;
Remote Control Field = Immediate Control Field;
Jump to Next Program Address;
```

### 19.5.3.10 DJZ (Decrement and Jump if Zero)<sup>†</sup>

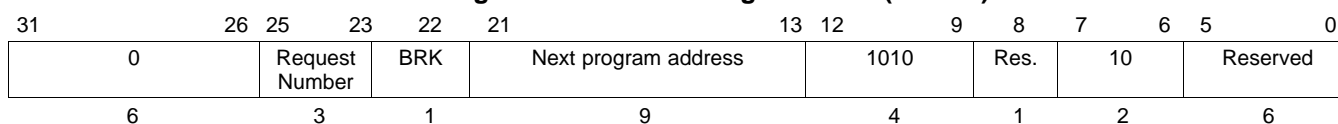
**Syntax**

```

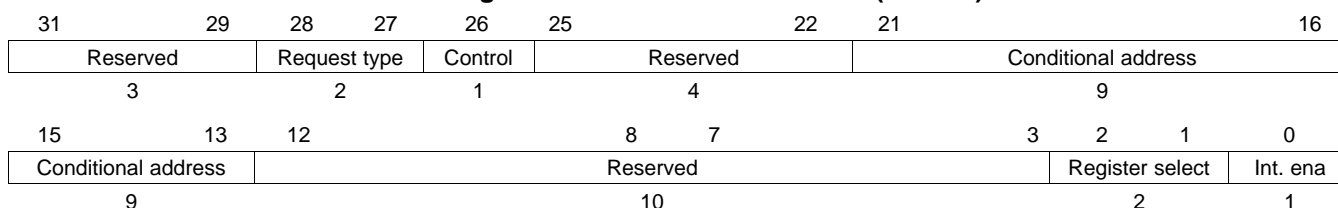
DJZ {
  [brk={OFF | ON}]
  [next={label | 9-bit unsigned integer}]
  [reqnum={3-bit unsigned integer}]
  [request={NOREQ | GENREQ | QUIET}]
  [control={OFF | ON}]
  [cond_addr={label | 9-bit unsigned integer}]
  [reg={A | B | T | NONE}]
  [irq={OFF | ON}]
  [data={25-bit unsigned integer}]
}

```

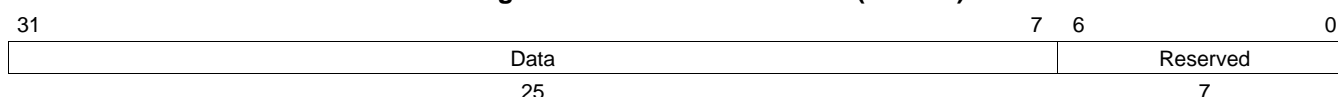
**Figure 19-101. DJZ Program Field (P31:P0)**



**Figure 19-102. DJZ Control Field (C31:C0)**



**Figure 19-103. DJZ Data Field (D31:D0)**



**Cycles** One

**Register modified** Selected register (A, B, or T)

<sup>†</sup> DJNZ is also supported syntax. The functionality of the two instruction names is identical.

**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.

<b>cond_addr</b>	This field is not optional for the DJZ instruction.
<b>irq</b>	ON generates an interrupt when the data field reaches zero. No interrupt is generated when the bit is OFF. Default: OFF.
<b>data</b>	Specifies the 25-bit integer value used as a counter. This counter is decremented each time the DJZ instruction is executed until the counter reaches 0. Default: 0.

### Execution

```

If (Data != 0)
{
    Data = Selected register = Data - 1;
    Jump to Next Program Address;
}
else
{
    Selected register = 000000h;

    If (Interrupt Enable == 1) SW interrupt flag = 1;
    If ([C28:C27] == 01) Generate request on request line [P25:P23];
    If ([C28:C27] == 11) Generate quiet request on request line [P25:P23];

    Jump to conditional Address;
}

```

### 19.5.3.11 ECMP (Equality Compare)

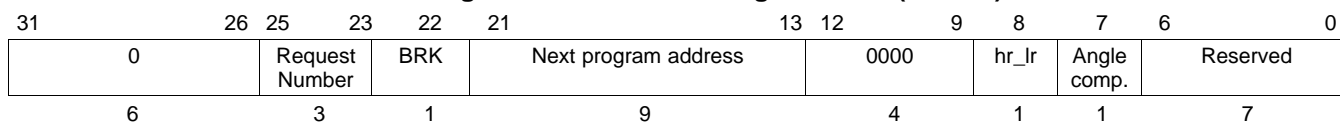
**Syntax**

```

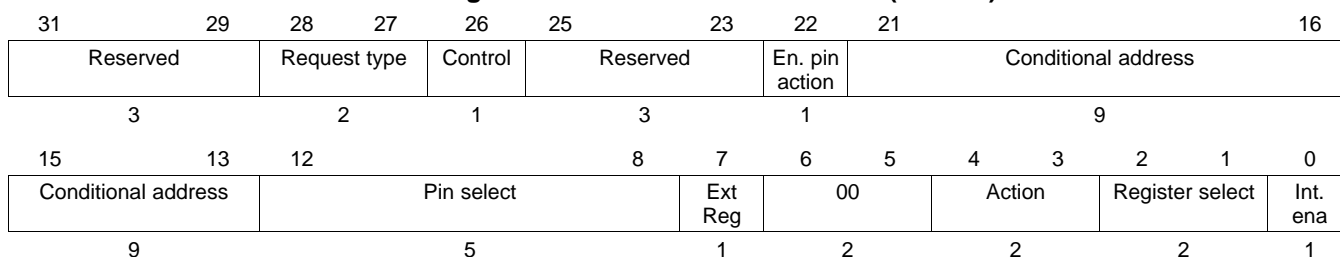
ECMP {
  [brk={OFF | ON}]
  [next={label | 9-bit unsigned integer}]
  [reqnum={3-bit unsigned integer}]
  [request={NOREQ | GENREQ | QUIET}]
  [hr_lr={HIGH | LOW}]
  [angle_comp={OFF | ON}]
  [control={OFF | ON}]
  [en_pin_action={OFF | ON}]
  [cond_addr={label | 9-bit unsigned integer}]
  pin={pin number}
  [action={CLEAR | SET | PULSELO | PULSEHI}]
  [reg={A | B | R | S | T | NONE}]
  [irq={OFF | ON}]
  [data={25-bit unsigned integer}]
  [hr_data={7-bit unsigned integer}]
}

```

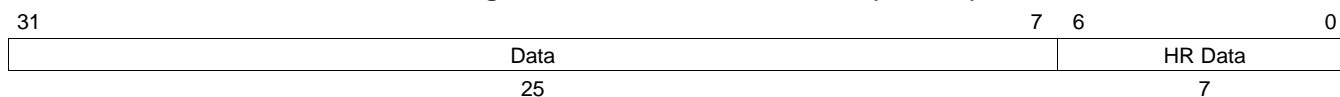
**Figure 19-104. ECMP Program Field (P31:P0)**



**Figure 19-105. ECMP Control Field (C31:C0)**



**Figure 19-106. ECMP Data Field (D31:D0)**



**Cycles** One

**Register modified** Register A, B, R, S or T if selected

**Description** ECMP can use all pins. This instruction compares a 25-bit data value stored in the data field (D31–D7) to the value stored in the selected ALU register (A, B, R, S, or T). Register select encoding can be found in [Section 19.5.2](#).

If R, S, or T registers are selected, and if the 25-bit data field matches, ECMP updates the register with the 32-bit value (D31-D0).

If the hr\_lr bit is cleared, the pin action will occur after a high resolution delay from the next resolution clock. If the hr\_lr bit is set, the delay is ignored. This delay is programmed in the data field (D6–D0).

The behavior of the pins is governed by the four action options in bits C4:C3. ECMP uses the zero flag to generate opposite pin action (synchronized to the loop resolution clock).

<b>angle_comp</b>	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.
<b>irq</b>	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.
<b>data</b>	Specifies the value for the data field. This value is compared with the selected register.
<b>hr_data</b>	Specifies the HR delay. Default: 0.

## Execution

```

If (Angle Comp. bit == 0 OR (Angle Comp. bit == 1 AND NAF_global == 1))
{
    If (Selected register value == Immediate data field value)
    {
        If (hr_lr bit == 0)
        {
            If (Enable Pin action == 1)
            {
                Selected Pin = Pin Action AT next loop resolution clock + HR delay;
            }
        }
        else
        {
            If (Enable Pin action == 1)
            {
                Selected Pin = Pin Action AT next loop resolution clock;
            }
        }

        If (Z == 1 AND Opposite action == 1)
        {
            If (Enable Pin action == 1)
            {
                Selected Pin = opposite Pin Action AT next loop resolution clock;
            }
        }

        If (Interrupt Enable == 1) SW interrupt flag = 1;
        If ([C28:C27] == 01) Generate request on request line [P25:P23];
        If ([C28:C27] == 11) Generate quiet request on request line [P25:P23];

        If (register R is selected) R register = Compare value (32 bit);
    }
}

```

```
        If (register S is selected) S register = Compare value (32 bit);
        If (register T is selected) T register = Compare value (32 bit);

        Jump to Conditional Address;
    }
}
elseif (Z == 1 AND Opposite action == 1)
{
    If (Enable Pin action == 1)
    {
        Selected Pin = opposite Pin Action AT next loop resolution clock;
    }
    Jump to Next Program Address;
}
else // Angle Comp. bit == 1 AND NAF_global == 0
{
    Jump to Next Program Address;
}
```

19.5.3.12 ECNT (Event Count)

**Syntax**

```

ECNT {
  [brk={OFF | ON}]
  [next={label | 9-bit unsigned integer}]
  [reqnum={3-bit unsigned integer}]
  [request={NOREQ | GENREQ | QUIET}]
  [control={OFF | ON}]
  [prv={OFF | ON}]
  [cond_addr={label | 9-bit unsigned integer}]
  pin={pin number}
  event={NAF | FALL | RISE | BOTH | ACCUHIGH | ACCULOW}
  [reg={A | B | R | S | T | NONE}]
  [irq={OFF | ON}]
  [data={25-bit unsigned integer}]
}

```

Figure 19-107. ECNT Program Field (P31:P0)

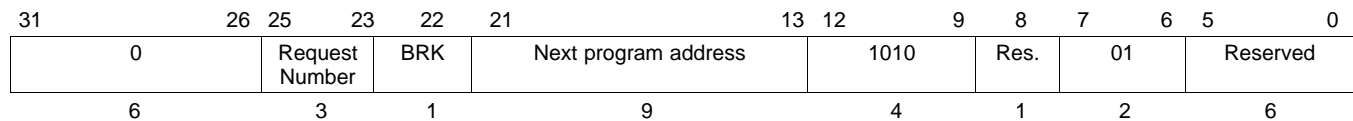


Figure 19-108. ECNT Control Field (C31:C0)

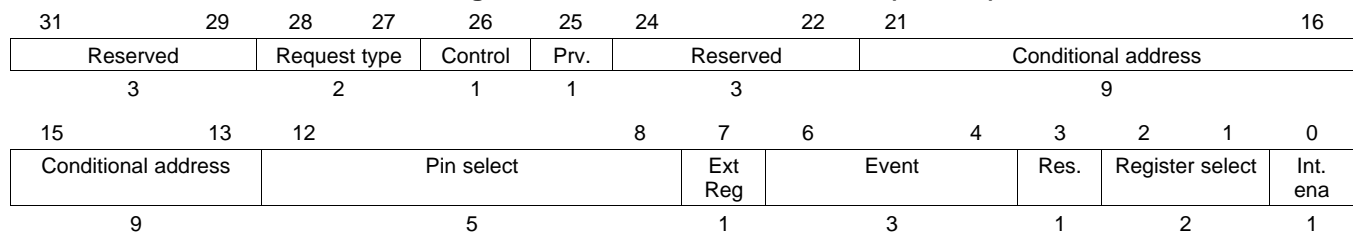
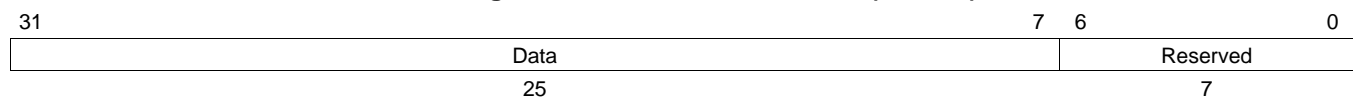


Figure 19-109. ECNT Data Field (D31:D0)



**Cycles** One cycle

**Register modified** Selected Register (A, B, R, S, T or none)

**Description** This instruction defines a specialized 25-bit virtual counter used as an event counter or pulse accumulator (see [Table 19-60](#)). The counter value is stored in the data field [D31:D7] and the selected register. If one of the 32-bit registers (R,S,T) is selected, the 25 bit count value is stored left justified in the register with zeros in the seven least significant bits.

When an event count condition is specified, the counter value is incremented on a pin edge condition or on the NAF condition (NAF is defined in ACNT). This instruction can be used with all pins.



**event** The event that triggers the counter.

**Table 19-60. Event Encoding Format for ECNT**

event	C6	C5	C4	Count Conditions	Mode	Int. Available
NAF	0	0	0	NAF flag is Set	Angle counter	Y
FALL	0	0	1	Falling edge on selected pin	Event counter	Y
RISE	0	1	0	Rising edge on selected pin	Event counter	Y
BOTH	0	1	1	Rising and Falling edge on selected pin	Event counter	Y
ACCUHIGH	1	0	-	while pin is high level	Pulse accumulation	N
ACCULOW	1	1	-	while pin is low level	Pulse accumulation	N

**irq** ON generates an interrupt when event in counter mode occurs. No interrupt is generated with OFF.  
Default: OFF.

**data** 25-bit integer value serving as a counter.  
Default: 0.

### Execution

```

If (event occurs)
{
  If (Register A or B Selected) {
    Selected register = Immediate Data Field + 1;
  }

  If (Register R, S or T Selected)
  {
    Selected register[31:7] = Immediate Data Field + 1;
    Selected register[6:0] = 0;
  }

  Immediate Data Field = Immediate Data Field + 1;

  If (Interrupt Enable == 1) SW interrupt flag = 1;
  If ([C28:C27] == 01) Generate request on line [P25:P23];
  If ([C28:C27] == 11) Generate quiet request on line [P25:P23];

  Jump to Conditional Address;
}
else
{
  Jump to Next Program Address;
}

Prv bit = Current Logic (Lx) value of selected pin; (Always executed)

```

19.5.3.13 MCMP (Magnitude Compare)

```

Syntax
MCMP {
  [brk={OFF | ON}]
  [next={label | 9-bit unsigned integer}]
  [reqnum={3-bit unsigned integer}]
  [request={NOREQ | GENREQ | QUIET}]
  [hr_lr={LOW | HIGH}]
  [angle_comp={OFF | ON}]
  [savesub={OFF | ON}]
  [control={OFF | ON}]
  [en_pin_action={OFF | ON}]
  [cond_addr={label | 9-bit unsigned integer}]
  pin={pin number}
  order={REG_GE_DATA | DATA_GE_REG}
  [action={CLEAR | SET | PULSELO | PULSEHI}]
  reg={A | B | R | S | T | NONE}
  [irq={OFF | ON}]
  [data={25-bit unsigned integer}]
  [hr_data={7-bit unsigned integer}]
}

```

Figure 19-110. MCMP Program Field (P31:P0)

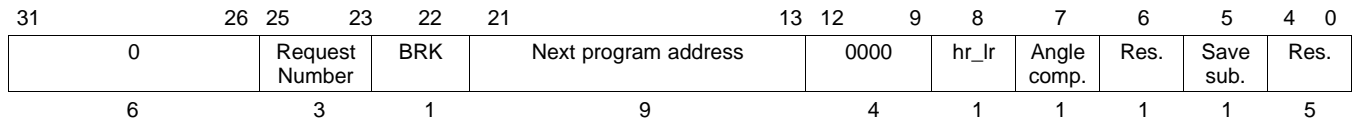


Figure 19-111. MCMP Control Field (C31:C0)

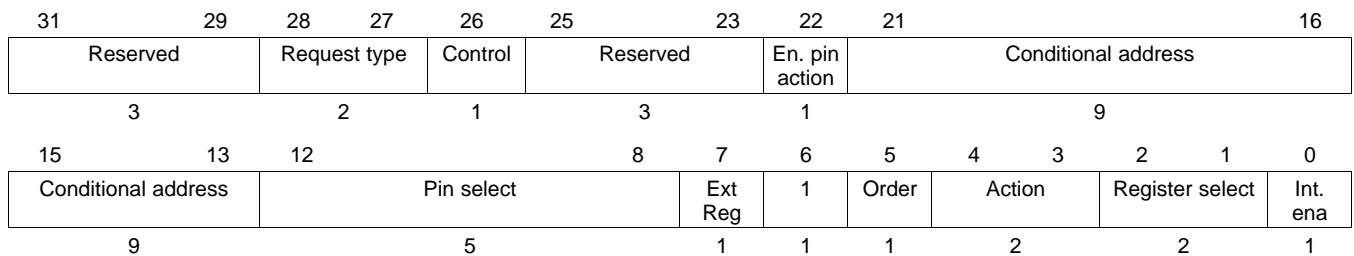
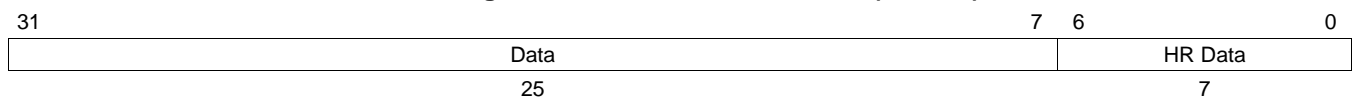


Figure 19-112. MCMP Data Field (D31:D0)



<b>Cycles</b>	One
<b>Register modified</b>	T (if save sub bit P[5] is set)
<b>Description</b>	<p>This instruction compares the magnitude of the 25-bit data value stored in the data field (D31-D7) and the 25-bit value stored in the selected ALU register (A, B, R, S, or T).</p> <p>If 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 (D6-D0).</p> <p>When the data value matches, an output pin can be set or reset according to the pin action bit (C[4]). The pin will not change states if the enable pin action bit (C[22]) is reset.</p> <p>MCMP uses the zero flag set to generate opposite pin action (synchronized to the loop resolution clock). The save sub bit (P[5]) provides the option to save the result of a subtraction into register T.</p>

---

**NOTE: The Difference Between Compare Values**

The difference between the two data values must not exceed  $(2^{24}) - 1$ .

---

<b>angle_comp</b>	<p>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.</p> <p>Default: OFF.</p>
<b>savesub</b>	<p>When set, the comparison result is saved into the T register (upper 25 bits).</p> <p>Default: OFF.</p>
<b>order</b>	Specifies the order of the operands for the comparison.

**Table 19-61. 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.

<b>irq</b>	Specifies whether or not an interrupt is generated. A value of ON sends an interrupt if the compare match occurs according to the order selected. If OFF is selected, no interrupt is generated.
<b>data</b>	Specifies the value for the data field. This value is compared with the selected register.
<b>hr_data</b>	HR delay. The default value for an unspecified bit is 0.

**Execution**

```

If (Angle Compare P[7] == 0 OR (P[7] == 1 AND NAF_global == 1))
{
    If( (Order C[5] == 1) AND (Data[31:7]- Selected register[31:7]) >= 0))
    OR ( (Order C[5] == 0) AND Selected register[31:7] - Data[31:7]) >= 0))
    {
        If (Order C[5] == 1 AND Save subtract P[5] == 1)
        {
            Register T[31:7] = Data[31:7] - Selected register[31:7];
            Register T[6:0] = 0;
        }

        If (Order C[5] == 0 AND Save subtract P[5] == 1)
        {
            Register T[31:7] = Selected register[31:7] - Data[31:7];
            Register T[6:0] = 0;
        }

        If (Enable Pin Action C[22] == 1)
        {
            If (hr_lr P[8] = 0) {
                Schedule Action on Selected Pin C[12:8] at start of next loop
                + HR Delay D[6:0];
            }
            else
            {
                Schedule Pin Action on Selected Pin C[12:8] at start of next loop;
            }
        }

        If (Interrupt Enable == 1) SW interrupt flag = 1;
        If ([C28:C27] == 01) Generate request on request line [P25:P23];
        If ([C28:C27] == 11) Generate quiet request on request line [P25:P23];

        Jump to Conditional Address;
    }
    else if (Z == 1 AND Opposite Action C[3] == 1 )
    {
        If (Enable Pin Action C[22] == 1)
        {
            Schedule Opposite Pin Action on Selected Pin C[12:8] at start of next loop;
        }

        Jump to Next Program Address;
    }
    else
        Jump to Next Program Address;
}
else // Angle Comp. bit == 1 AND NAF_global == 0
    Jump to Next Program Address;

```

### 19.5.3.14 MOV32 (MOVE 32)

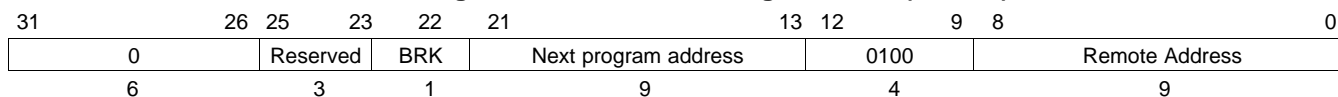
**Syntax**

```

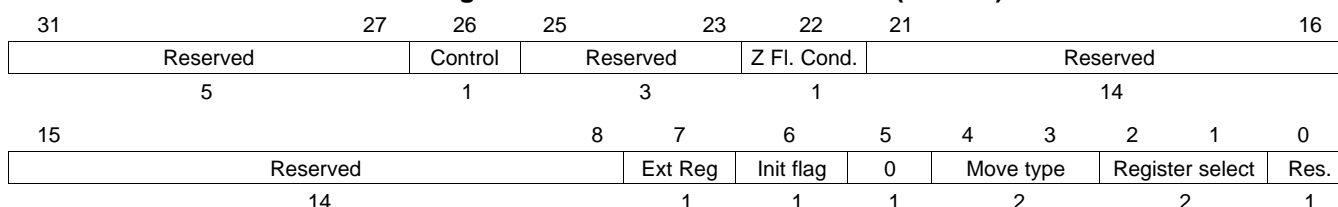
MOV32 {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
[remote={label | 9-bit unsigned integer}]
[control={OFF | ON}]
[z_cond={OFF | ON}]
[init={OFF | ON}] ON}}
type={IMTOREG | IMTOREG&REM | REGTOREM | REMTOREG}
[reg={A | B | R | S | T | NONE}]
[data={25-bit unsigned integer}]
[hr_data={7-bit unsigned integer}]
}

```

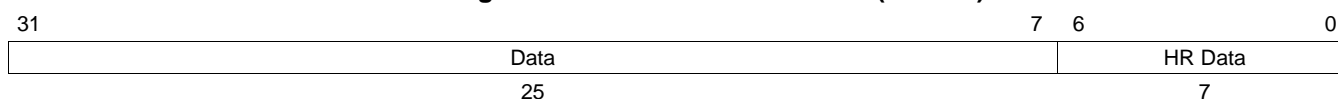
**Figure 19-113. MOV32 Program Field (P31:P0)**



**Figure 19-114. MOV32 Control Field (C31:C0)**



**Figure 19-115. MOV32 Data Field (D31:D0)**



**Cycles** One or two cycles

**Register modified** Selected register (A, B, R, S, or T)

**Description** MOV32 replaces the selected ALU register and/or the data field values at the remote address location depending on the move type. [Figure 19-116](#) through [Figure 19-119](#) 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.

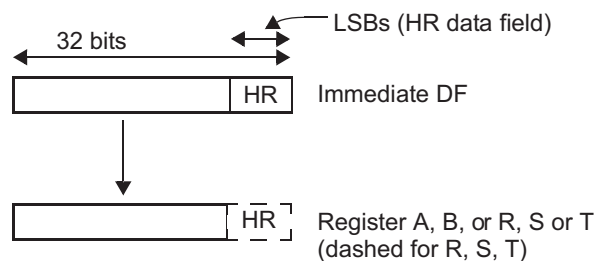
<b>remote</b>	Determines the location of the remote address. Default: Current instruction + 1.
<b>z_cond</b>	When set to OFF the MOV32 performs the move operation specified by the move type whenever it is executed (independent on the state of the Z-Flag). When set to ON the MOV32 performs the move operation specified by the move type only when the Z-Flag is set.
<b>init</b>	(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.
<b>type</b>	Specifies the move type to be executed.

**Table 19-62. Move Type Encoding Selection**

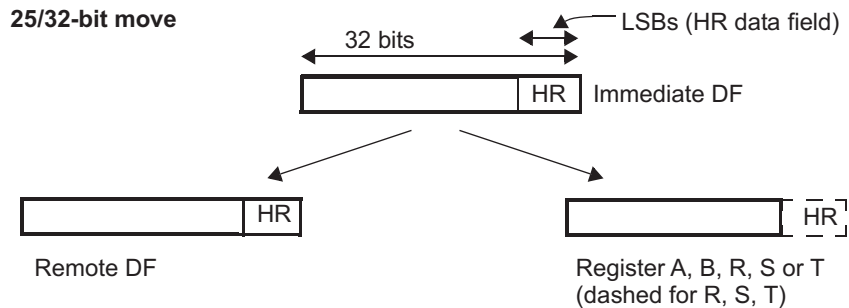
Move Type	C4	C3	Source	Destination(s)	Cycles
IMTOREG	0	0	Immediate data field	Register A, B, or T	1
IMTOREG&REM	0	1	Immediate data field	Remote data field & register A, B, or T	1
REGTOREM	1	0	Register A, B, or T	Remote data field	1
REMTOREG	1	1	Remote data field	Register A, B, or T	2

**Figure 19-116. MOV32 Move Operation for IMTOREG (Case 00)**

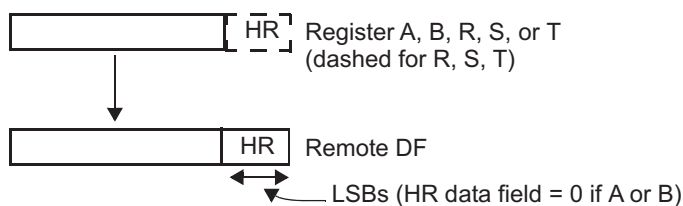
25/32-bit move



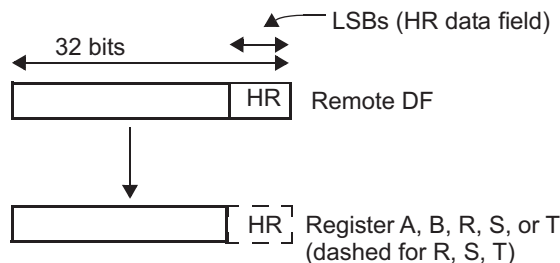
<b>reg</b>	Specifies which register (A, B, T, or NONE) is involved in the move. A register (A, B, or T) must be specified for every move type except IMTOREG&REM. If <i>NONE</i> is used with move type IMTOREG&REM, the MOV32 executes a move from the immediate data field to the remote data field. If <i>NONE</i> is used with any other move type, no move is executed.
<b>data</b>	Specifies a 25-bit integer value to be written to the remote data field or selected register.
<b>hr_data</b>	(Optional) HR delay. The default value for an unspecified bit is 0.

**Figure 19-117. MOV32 Move Operation for IMTOREG&REM (Case 01)****Figure 19-118. MOV32 Move Operation for REGTOREM (Case 10)**

25/32-bit move

**Figure 19-119. MOV32 Move Operation for REMTOREG (Case 11)**

25/32-bit move

**Execution**

```

If [(z_cond C[22] == 0) OR ((z_cond C[22] == 1) AND (Z Flag == 1))]
{
  switch (type C[4:3])
  {
    case 00: // IMTOREG
      Selected register = Immediate Data Field;
    case 01: // IMTOREG&REM
      Selected register = Immediate Data Field;
      Remote Data Field = Immediate Data Field;
    case 10: // REGTOREM
      Remote Data Field = Selected register;
    case 11: // REMTOREG
      Selected register = Remote Data Field;
  }
}

If (Init Flag == 1)
{
  ACF = 0;
  DCF = 1;
  GPF = 0;
  NAF = 0;
}
else

```

All flags remain unchanged;

Jump to Next Program Address;



### 19.5.3.15 MOV64 (Data Move)

**Syntax**

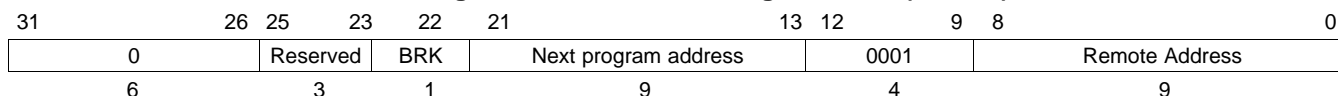
```
MOV64 {
  [brk={OFF | ON}]
  [next={label | 9-bit unsigned integer}]
  remote={label | 9-bit unsigned integer}
  [request={NOREQ | GENREQ | QUIET}]
  [control={OFF | ON}]
  [en_pin_action={OFF | ON}]
  [cond_addr={label | 9-bit unsigned integer}]
  [pin={pin number}]
  comp_mode={ECMP | SCMP | MCMP1 | MCMP2}
  [action={CLEAR | SET | PULSELO | PULSEHI}]
  [reg={A | B | R | S | T | NONE}]
  [irq={OFF | ON}]
  [data={25-bit unsigned integer}]
  [hr_data= {7-bit unsigned integer}]
}
```

-or-

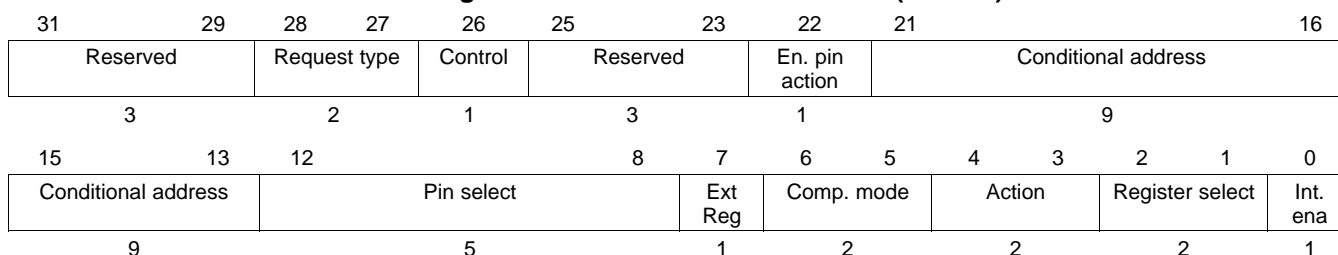
**Syntax**

```
MOV64 {
  [brk={OFF | ON}]
  [next={label | 9-bit unsigned integer}]
  remote={label | 9-bit unsigned integer}
  cntl_val={29-bit unsigned integer}
  [data={25-bit unsigned integer}]
  [hr_data= {7-bit unsigned integer}]
}
```

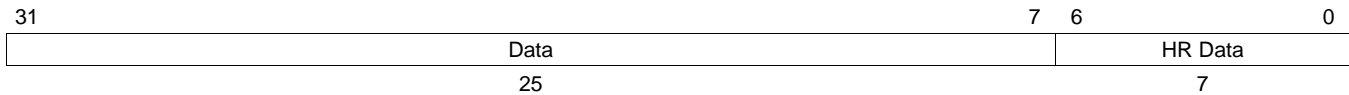
**Figure 19-120. MOV64 Program Field (P31:P0)**



**Figure 19-121. MOV64 Control Field (C31:C0)**



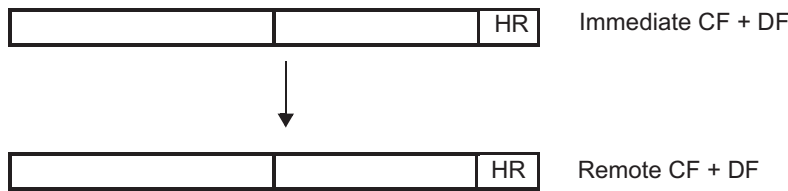
**Figure 19-122. MOV64 Data Field (D31:D0)**



**Cycles** One  
**Register modified** None

**Description** This instruction modifies the data field and the control field at the remote address.  
 MOV64 has two distinct syntaxes. In the first syntax, bit values may be set by assigning a value to each of the control fields. This syntax is convenient for modifying control fields that are arranged similarly to the format of the MOV64 control field. A second syntax, in which the entire 29-bit control field is specified by the cntl\_val field, is convenient when the remote control field is dissimilar to the MOV64 control field. Either syntax may be used, but you must use one or the other but not a combination of syntaxes. See [Figure 19-123](#).

**Figure 19-123. MOV64 Move Operation**



**Table 19-63. MOV64 Control Field Descriptions**

request	Maintains the control field for the remote instruction.
control	Maintains the control field for the remote instruction.
en_pin_action	Maintains the control field for the remote instruction.
cond_addr	Maintains the control field for the remote instruction.
pin	Maintains the control field for the remote instruction.
register, ext reg	Maintains the control field for the remote instruction.
comp_mode	Selects the comparison mode type to be used by the remote instruction.
action	Maintains the control field for the remote instruction.
irq	Maintains the control field for the remote instruction.
data	Specifies the 25-bit initial count value for the data field. If omitted, the field defaults to 0.
hr_data	(Optional) HR delay. The default value for an unspecified bit is 0.

**Table 19-64. Comparison Type Encoding Format**

comp_mode	C[6]	C[5]	MCMP order
ECMP	0	0	
SCMP	0	1	
MCMP1	1	0	REG_GE_DATA
MCMP2	1	1	DATA_GE_REG

### Execution

Remote Data Field = Immediate Data Field;  
Remote Control Field = Immediate control Field;  
Jump to Next Program Address;

19.5.3.16 PCNT (Period/Pulse Count)

**Syntax**

```

PCNT {
  [hr_lr={HIGH | LOW}]
  [brk={OFF | ON}]
  [next={label | 9-bit unsigned integer}]
  [reqnum={3-bit unsigned integer}]
  [request={NOREQ | GENREQ | QUIET}]
  [irq={OFF | ON}]
  type={FALL2RISE | RISE2FALL | FALL2FALL | RISE2RISE}
  pin={pin number}
  [control={OFF | ON}]
  [prv={OFF | ON}]
  [period={25-bit unsigned integer}]
  [data={25-bit unsigned integer}]
  [hr_data= {7-bit unsigned integer}]
}

```

Figure 19-124. PCNT Program Field (P31:P0)

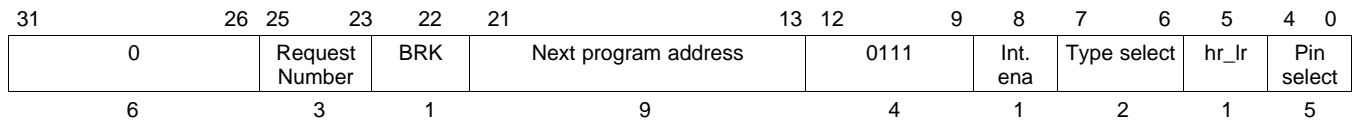


Figure 19-125. PCNT Control Field (C31:C0)

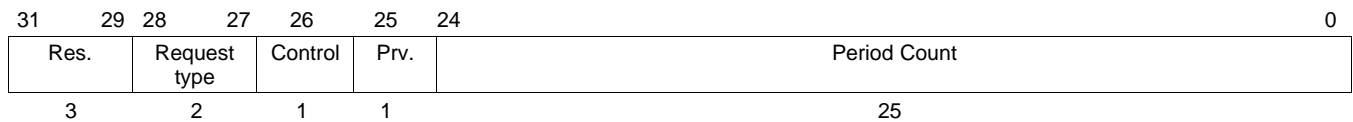
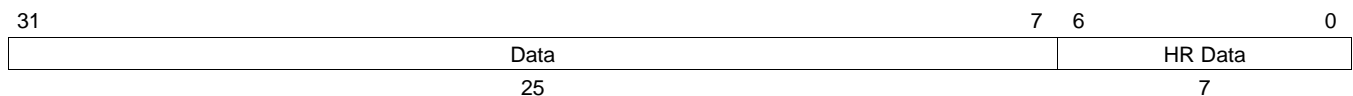


Figure 19-126. PCNT Data Field (D31:D0)



**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 C[24:0] and in the register A is incremented each N2HET loop. PCNT uses the HR structure on the pin to measure an HR period/pulse count value.

**hr\_lr** (Optional) Specifies whether the PCNT instruction captures the HR delay into the HR data field on the selected edge condition. If hr\_lr is 0 (HIGH) then PCNT captures the HR delay. if hr\_lr is 1 (LOW) then PCNT only captures at loop resolution.

<b>irq</b>	(Optional) Specifies whether or not an interrupt is generated. A value of ON sends an interrupt when a new value is captured. If OFF is selected, no interrupt is generated.
<b>type</b>	(Optional) Determines the type of counter that is implemented.

**Table 19-65. Counter Type Encoding Format**

	P7	P6	Period/Pulse Select	Reset On	Capture On
FALL2RISE	0	0	Count low pulse duration on selected pin	Falling edge	Rising edge
RISE2FALL	0	1	Count high pulse duration on selected pin	Rising edge	Falling edge
FALL2FALL	1	0	Count period between falling edges on selected pin	Falling edge	Falling edge
RISE2RISE	1	1	Count period between rising edges on selected pin	Rising edge	Rising edge

<b>period</b>	Specifies the 25-bit integer value that holds the counter value. The counter value is also stored in register A. Default: 0.
<b>data</b>	25-bit integer representing the last captured counter value. Default: 0.
<b>hr_data</b>	HR delay. Default: 0.

If *period-measure* is selected, PCNT captures the counter value into the period/pulse data field [D31:D7] on the selected edge. The HR structure provides HR capture field [D6:D0]. The counter value [C24:C0] is reset on the same edge. The captured period value is a 32-bit value.

If *pulse-measure* is selected, PCNT captures the counter value into the period/pulse count field [D31:D7] on the selected edge. The HR structure provides HR capture field [D6:D0]. The counter value [C24:C0] is reset on the next opposite edge. The captured pulse value is a 32-bit value.

When the overflow count (all 1's in the counter value) is reached, PCNT stops counting until the next reset edge is detected.

Note: For FALL2FALL/RISE2RISE, the user should always discard the first interrupt/HTU request if interrupt/request are enabled before HET\_ON. For both the types, reset edge and capture edge are the same and the interrupt or HTU request is triggered on capture edge (which is nothing but the reset edge). Once the execution unit is enabled, the first edge generates an interrupt but the value of the counter is of no use as this is not the period between 2 edges. So first edge after turning on N2HET is used mainly for resetting the counter and start the period count.

### Execution

```

Z = 0;

If (Period C[24:0] != 1FF_FFFFh) {
    Period C[24:0] = Period C[24:0] + 1;
}

Register A = Period C[24:0];

If (specified capture edge detected on selected pin)
{
    Z = 1;
}

```

```
    If (Period value != 1FF_FFFFh)
    {
        HR Capture Value = selected HR counter;
    }
    else
    {
        HR Capture Value = 7Fh;
    }

    If (Interrupt Enable == 1) SW interrupt flag = 1;
    If ([C28:C27] == 01) Generate request on request line [P25:P23];
    If ([C28:C27] == 11) Generate quiet request on request line [P25:P23];
}

If (specified reset edge detected on selected pin)
{
    Period value = 0000000h;
}

Prv bit = Current Logic (Lx) value of selected pin;

Jump to Next Program Address;
```

### 19.5.3.17 PWCNT (Pulse Width Count)

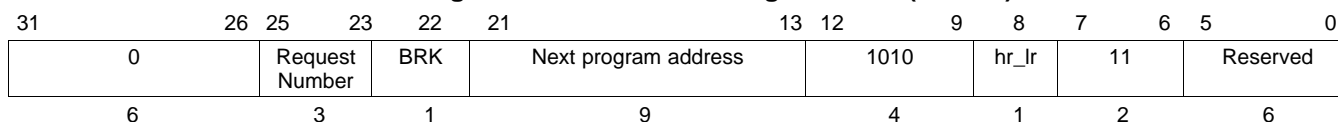
**Syntax**

```

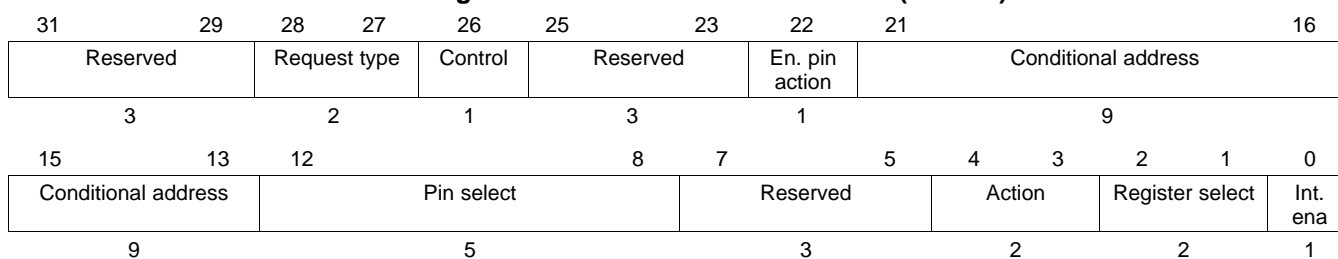
PWCNT {
  [brk={OFF | ON}]
  [next={label | 9-bit unsigned integer}]
  [reqnum={3-bit unsigned integer}]
  [request={NOREQ | GENREQ | QUIET}]
  [hr_lr={HIGH | LOW}]
  [control={OFF | ON}]
  [cond_addr={label | 9-bit unsigned integer}]
  [en_pin_action={OFF | ON}]
  pin = {pin number}
  [action={CLEAR | SET | PULSELO | PULSEHI}]
  [reg={A | B | T | NONE}]
  [irq={OFF | ON}]
  [data={25-bit unsigned integer}]
  [hr_data={7-bit unsigned integer}]
}

```

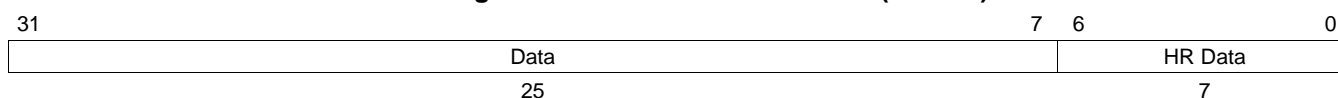
**Figure 19-127. PWCNT Program Field (P31:P0)**



**Figure 19-128. PWCNT Control Field (C31:C0)**



**Figure 19-129. PWCNT Data Field (D31:D0)**



**Cycles** One

**Register modified** Selected register (A, B or T)

**Description** This instruction defines a virtual timer used to generate variable length pulses. The counter value stored in the data field is decremented unconditionally on each timer resolution until it reaches zero, and it then stays at zero until it is reloaded with a non-zero value.

The specified pin action is performed as long as the count after count value is decremented is greater than 0. The opposite pin action is performed when the count after decrement just reaches 0.

If the hr\_lr bit is reset, the opposite pin action will be taken after a HR delay from the next resolution clock. If the hr\_lr bit is set, the delay is ignored. This delay is programmed in bits [D6:D0].

<b>irq</b>	ON generates an interrupt when the data field value reaches 0. No interrupt is generated for OFF. Default: OFF.
<b>data</b>	25-bit integer value serving as a counter.
<b>hr_data</b>	HR delay. Default: 0.

### Execution

```

If (Data field value == 0)
{
    Selected register = 0;
    Jump to Next Program Address;
}

If (Data field value > 1)
{
    Selected register = Data field value - 1;
    Data field value = Counter value - 1;

    If (Enable Pin action == 1)
    {
        Selected Pin = Pin Action AT next loop resolution clock;
    }

    Jump to Next Program Address;
}

If (Data field value == 1)
{
    Selected register = 0000000h;
    Data field value = 0000000h;

    If (Opposite action == 1)
    {
        If (hr_lr bit == 0)
        {
            If (Enable Pin action == 1)
            {
                Selected Pin = Opposite level of Pin Action AT next loop resolution clock
                + HR delay;
            }
        }
        else
        {
            If (Enable Pin action == 1)
            {
                Selected Pin = Opposite level of Pin Action AT next loop
                resolution clock;
            }
        }
    }

    If (Interrupt Enable == 1) SW interrupt flag = 1;
    IF ([C28:C27] == 01) Generate request on request line [P25:P23];
}

```



---

```
    If ([C28:C27] == 11) Generate quiet request on request line [P25:P23];  
    }  
    Jump to Conditional Address  
}
```

19.5.3.18 RADM64 (Register Add Move)

**Syntax**

```

RADM64 {
  [brk={OFF | ON}]
  [next={label | 9-bit unsigned integer}]
  remote={label | 9-bit unsigned integer}
  [request={NOREQ | GENREQ | QUIET}]
  [control={OFF | ON}]
  [en_pin_action={OFF | ON}]
  [cond_addr={label | 9-bit unsigned integer}]
  [pin={pin number}]
  comp_mode={ECMP | SCMP | MCMP1 | MCMP2}
  [action={CLEAR | SET | PULSELO | PULSEHI}]
  [reg={A | B | R | S | T | NONE}]
  [irq={OFF | ON}]
  [data={25-bit unsigned integer}]
  [hr_data= {7-bit unsigned integer}]
}

```

-or-

**Syntax**

```

RADM64 {
  [brk={OFF | ON}]
  [next={label | 9-bit unsigned integer}]
  remote={label | 9-bit unsigned integer}
  cntl_val={29-bit unsigned integer}
  [data={25-bit unsigned integer}]
  [hr_data= {7-bit unsigned integer}]
}

```

Figure 19-130. RADM64 Program Field (P31:P0)

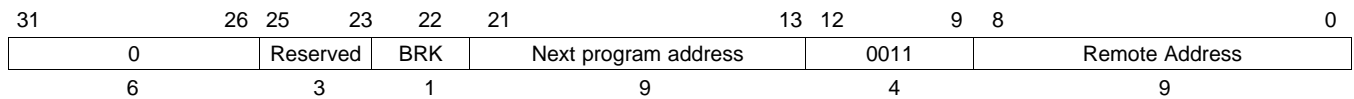
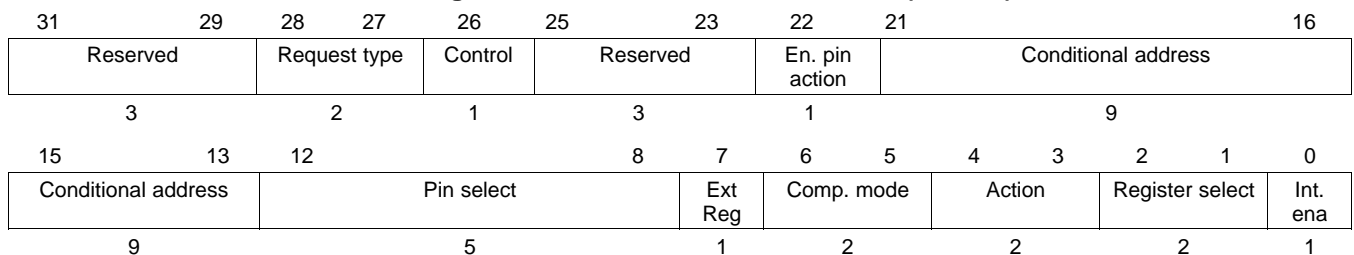
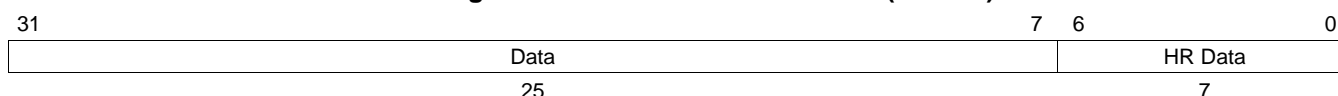


Figure 19-131. RADM64 Control Field (C31:C0)



**Figure 19-132. RADM64 Data Field (D31:D0)**



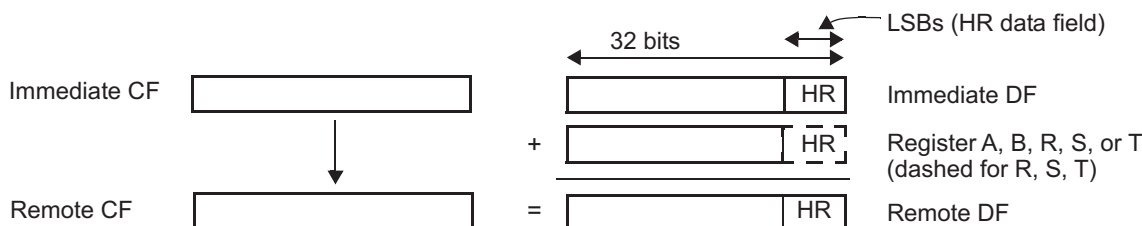
**Cycles** Normally One Cycle. Two cycles if writing to remote address that is also the next address.

**Register modified** Selected register (A, B, R, S, 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 the R, S, or T register is selected, the addition is a 32-bit addition. The table description shows the bit encoding for determining which ALU register is selected.

RADM64 has two distinct syntaxes. In the first syntax, bit values may be set by assigning a value to each of the control fields. This syntax is convenient for modifying control fields that are arranged similar to the format of the RADM64 control field. A second syntax, in which the entire 29-bit control field is specified by the `cntl_val` field, is convenient when the remote control field is dissimilar from the RADM64 control field. Either syntax may be used, but you must use one or the other but not a combination of syntaxes. See [Figure 19-133](#).

**Figure 19-133. RADM64 Add and Move Operation**



**comp\_mode** Selects the comparison mode type to be used.

**Table 19-66. Comparison Type Encoding Format**

comp_mode	C[6]	C[5]	MCMP order
ECMP	0	0	
SCMP	0	1	
MCMP1	1	0	REG_GE_DATA
MCMP2	1	1	DATA_GE_REG

**Table 19-67. RADM64 Control Field Descriptions**

request	Maintains the control field for the remote instruction.
Control	Maintains the control field for the remote instruction.
en_pin_action	Maintains the control field for the remote instruction.
cond_addr	Maintains the control field for the remote instruction.
pin	Maintains the control field for the remote instruction.
register	Maintains the control field for the remote instruction.
action	Maintains the control field for the remote instruction.

**Table 19-67. RADM64 Control Field Descriptions (continued)**

irq	Maintains the control field for the remote instruction.
data	Specifies the 25-bit initial value for the data field. If omitted, the field defaults to 0.
hr_data	Seven least significant bits of the 32-bit data field. Default: 0.
cntl_val	Specifies the 29 least significant bits of the Control field.

**Execution**

Remote Data Field = Selected register + Immediate Data Field (including HR field);  
Remote Control Field = Immediate Control Field;  
Jump to Next Program Address;

### 19.5.3.19 RCNT (Ratio Count)

**Syntax**

```
RCNT {
  [brk={OFF | ON}]
  [next={label | 9-bit unsigned integer}]
  [control={OFF | ON}]
  divisor={25-bit unsigned integer}
  [data={25-bit unsigned integer}]
}
```

**Figure 19-134. RCNT Program Field (P31:P0)**

31	26	25	23	22	21	13	12	9	8	7	6	5	4	3	1	0			
0						Reserved			BRK	Next program address				1010	Res.	00	Step width	Res.	1
6						3			1	9				4	1	2	2	3	1

**Figure 19-135. RCNT Control Field (C31:C0)**

31	27	26	25	24	0
Reserved			Control	Res.	Divisor
5			1	1	25

**Figure 19-136. RCNT Data Field (D31:D0)**

31	7	6	0
Data			Reserved
25			7

**Cycles** Two Cycles (One Cycle if T=0)

**Register modified** None

**Description**

RCNT is used with other instructions to convert an input period measurement  $T_{\text{Input}}$  to the form of (Equation 30) where the input period is expressed as a fraction of a reference period  $T_{\text{Reference}}$ .

$$T_{\text{Input}} = T_{\text{Reference}} \cdot \left( \frac{N}{M} \right) \quad (30)$$

RCNT computes the numerator N of (Equation 30). The denominator M of (Equation 30) is a constant that is of interest. For example, choosing allows the input period to be expressed as a percentage (%) of the reference period. Note that if  $T_{\text{Input}} > T_{\text{Reference}}$ , then RCNT will return  $N > M$ ; which would be correct if for example the input pulse period is 110% of the reference pulse period.

RCNT expects that register T is loaded with the value of  $T_{\text{Reference}}$ . The input period  $T_{\text{Input}}$  is determined by counting the number of loop resolution periods between edges on the input pin. This information is conveyed through the Z flag from a PCNT instruction that precedes the RCNT instruction.

The divisor field of the RCNT instruction should be chosen as:  
 $\text{Divisor} = M \cdot \text{lr}$ , where M is the desired denominator from (Equation 30) and lr is the loop resolution prescale value.

An example N2HET program that makes use of the RCNT instruction is:

```
L0: MOV32 { remote=dummy,type=IMTOREG,reg=T,data=0x8,hr_data=0};

L1: PCNT { hr_lr=HIGH,brk=OFF,type=FALL2FALL,pin=0};

L2: RCNT { divisor=320,data=0x4};

L3: BR {cond_addr=L5, event = Z}

L4: ADC { src1=ZERO,src2=IMM,dest=IMM,next=L0,data=0,hr_data=0};

L5: ADD { src1=REM,src2=ZERO,dest=IMM,remote=L4,data=0,hr_data=0};

L6: ADD { src1=ZERO,src2=ZERO,dest=NONE,rdest=REM,
        next=L0,remote=L4,data=0,hr_data=0};
```

dummy

In this small program an input signal on pin 0 is measured both in terms of absolute cycles by the PCNT instruction at L1 and as in 1/10ths of the reference period by the RCNT instruction at L2. In this example the reference period is a constant 0x400 cycles; this value is loaded into register T by the MOV32 instruction at L0. (0x400 is data=8, hr\_data=0)

RCNT follows PCNT and is initialized to a working count of T/2 (0x200) whenever the PCNT instruction detects a falling edge on pin 0. Between falling edges on pin0, RCNT accumulates counts 10x faster than PCNT; so that the working data field of RCNT will reach the reference value of 0x400 in 1/10th the time that a PCNT instruction would. Each time the RCNT instruction passes the reference value, it sets the carry out flag and subtracts the reference value from the working count. By accumulating carry-outs from RCNT, the add with carry instruction at L4 effectively counts in increments of 1/10th of the reference period. Note that the divisor value 320 is 10 times 32; this assumes lr=32.

When the next falling edge is detected on pin 0, PCNT sets the Z flag and the RCNT instruction resets again to the initial data field of T/2. RCNT does not modify the Z flag, so that the branch instruction at L3 can execute instructions at L5, L6 instead of L4. The instructions at L5 and L6 capture the final result from L4 and reset the ADC instruction at L4 to zero for the start of the next period measurement.

### Execution

```
If (register T[31:0] != 00000000h)
{
    C = 0;

    If (Z == 0)
    {
        Data Field[31:0] = Data Field[31:0] + Divisor[24:0];

        If (Data Field[31:0] >= Reg T[31:0])
        {
            Data Field[31:0]=Data Field[31:0] - Reg T[31:0];
            C = 1;
        }
    }
    else
    {
        Data Field[31:0] = T[31:0] >> 1; /* T/2 */
    }
}
Jump to Next Program Address;
```

### 19.5.3.20 SCMP (Sequence Compare)

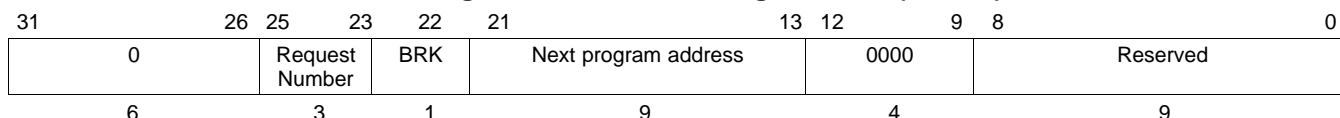
**Syntax**

```

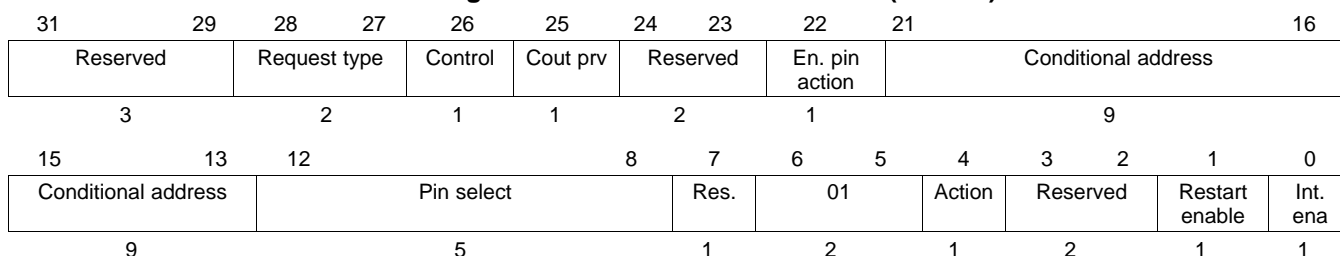
SCMP {
  [brk={OFF | ON}]
  [next={label | 9-bit unsigned integer}]
  [reqnum={3-bit unsigned integer}]
  [request={NOREQ | GENREQ | QUIET}]
  [control={OFF | ON}]
  [en_pin_action={OFF | ON}]
  cond_addr={label | 9-bit unsigned integer}
  pin = {pin number}
  [action={CLEAR | SET}]
  [restart={OFF | ON}]
  [irq={OFF | ON}]
  [data={25-bit unsigned integer}]
}

```

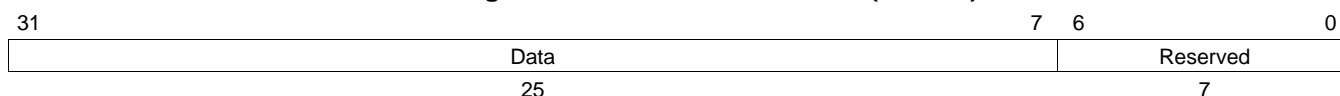
**Figure 19-137. SCMP Program Field (P31:P0)**



**Figure 19-138. SCMP Control Field (C31:C0)**



**Figure 19-139. 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 (C13) specifies whether the instruction is operating in angle or time operation mode.

When the compared values match in angle mode, a pin can be set or reset according to the pin action bit (C4). The pin does not change states if the enable pin action bit (C22) is reset.

The restart enable bit (C1) provides the option to unconditionally restart a sequence using the X-flag bit of ACMP.

<b>restart</b>	If restart is set to ON and the X flag = 1, the assembler writes a value of 1 into the immediate index field, writes the value in register A into the immediate data field, and jumps to the next program address. The X flag is set or cleared by the ACMP instruction. If restart is set to OFF, the X flag is ignored; no special action is performed. Default: OFF.
<b>irq</b>	ON generates an interrupt if the compare match occurs in angle mode. No interrupt is generated when the field is OFF. Default: OFF.
<b>data</b>	Specifies the 25-bit compare value.
<b>cond_addr</b>	Since the LSB of the conditional address is used to select between time mode and angle mode, and since the conditional address is taken only in time mode, the destination for the conditional address must be odd.

## Execution

```

If (Data field value <= Selected register value) Cout = 0; else Cout = 1;

If (Restart Enable == 1 AND X == 1)
{
    C13 = 1;
    Immediate Data Field = Register A;
    Cout = 0;
    Jump to Next Program Address;
}

If (Angle Mode (C13 == 0) AND ((Restart En. == 1 AND X == 0) OR Restart En. == 0))
{
    If (Z == 0 AND (Register B value - Angle Inc. < Data field value) AND Cout == 0) OR
        (Z == 1 AND (Cout_prv == 1 OR Cout == 0))
    {
        If (Enable Pin Action == 1) Selected Pin = Pin Action;
        If (Interrupt Enable == 1) SW interrupt flag = 1;
        If ([C28:C27] == 01) Generate request on request line [P25:P23];
        If ([C28:C27] == 11) Generate quiet request on request line [P25:P23];

        Immediate Data Field = Register A;
        C13 = 1; /** switch to Time Mode ***/
    }

    Jump to Next Program Address;
}
Else If (Time Mode (C13 == 1)) AND ((Restart En. == 1 AND X == 0) OR Restart En. == 0)
{
    /* Result of subtract must not exceed 2^24 - 1 */
    Register T = Register A - Immediate Data Field;
    Jump to Conditional Program Address;
}

Cout_prv = Cout; (always executed)

```



### 19.5.3.21 SCNT (Step Count)

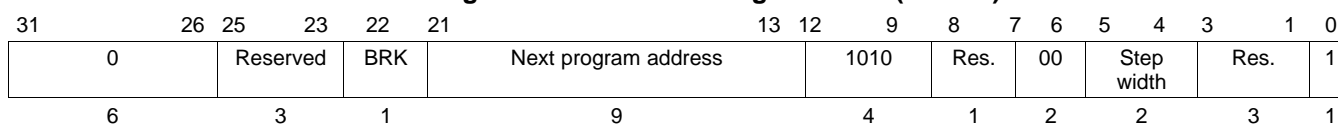
**Syntax**

```

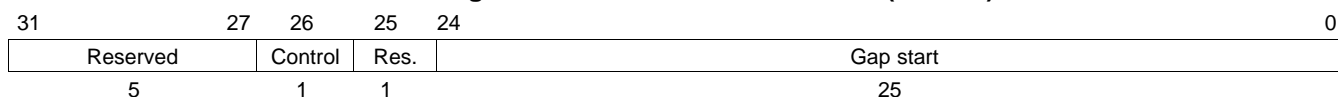
SCNT {
  [brk={OFF | ON}]
  [next={label | 9-bit unsigned integer}]
  step={8 | 16 | 32 | 64}
  [control={OFF | ON}]
  gapstart={25-bit unsigned integer}
  [data={25-bit unsigned integer}]
}

```

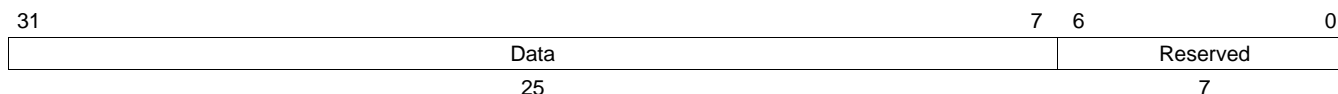
**Figure 19-140. SCNT Program Field (P31:P0)**



**Figure 19-141. SCNT Control Field (C31:C0)**



**Figure 19-142. SCNT Data Field (D31:D0)**



**Cycles** One or two cycles (two cycles when DF is involved in the calculations)

**Register modified** Register A

**Description** This instruction can be used only once in a program and defines a specialized virtual timer used after APCNT and before ACNT to generate an angle-referenced time base synchronized to an external signal (that is, a toothed wheel signal) as defined in APCNT and ACNT. Step width selection bits are saved in two flags, SWF0, and SWF1, to be re-used in ACNT. SCNT multiplies the frequency of the external signal by a constant  $K$  defined in the step width field, [P5:P4]. The bit encoding for this field is defined in [Table 19-68](#):

**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 19-68](#).

**Table 19-68. Step Width Encoding for SCNT**

P5	P4	Step Width (K)
0	0	8
0	1	16
1	0	32
1	1	64

<b>gapstart</b>	<p>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:</p> $GAPSTART = (stepwidth \times (actual\ teeth\ on\ gear - 1)) + 1.$
<b>data</b>	<p>Specifies the 25-bit integer value serving as a counter. Default: 0.</p>

This instruction is incremented by the step value K on each timer resolution up to the previous period value P(n-1) measured by APCNT (stored in register T). The resulting period of SCNT is: P(n - 1)/K

Due to stepping, the final count of SCNT will not usually exactly match the target p(n-1). SCNT compensates for this error by starting each cycle with the remainder of the previous cycle.

When SCNT reaches the target p(n-1), the zero flag is set as an increment condition for ACNT. SCNT also specifies a gap start angle, defining the start of a range in ACNT where period measurements in APCNT are temporarily stopped to mask singularities in the external signal.

SCNT uses register A to store the gap start value. Gap start has no effect for SCNT.

### Execution

```

SWF1 = P5;
SWF0 = P4;
Z = 0;

If (register T != 0000000h)
{
    If (DCF == 1 OR ACF == 1)
    {
        Data Field register = 0000000h;
        Counter value = 0000000h;
    }

    If (DCF == 0 AND ACF == 0)
    {
        Data Field register = Data field register + Step Width;
    }

    If ((Data Field register - register T) >= 0)
    {
        Data field register = Data Field register - register T;
        Z = 1;
    }

    Register A = Gap start value;
}

Jump to Next Program Address;

```

### 19.5.3.22 SHFT (Shift)

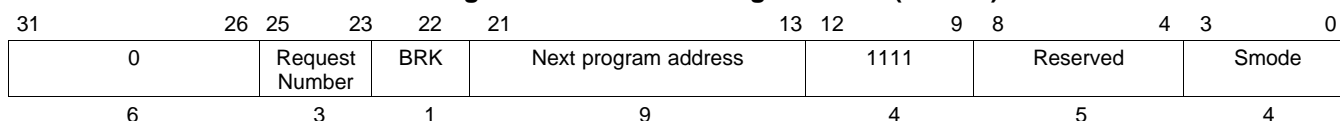
**Syntax**

```

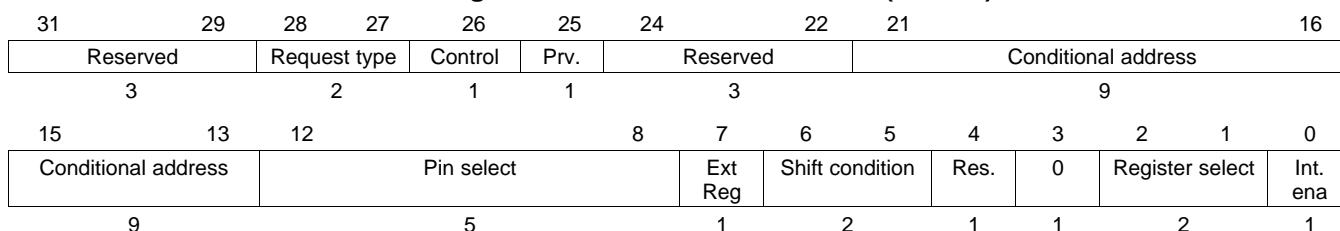
SHFT {
  [brk={OFF | ON}]
  [next={label | 9-bit unsigned integer}]
  [reqnum={3-bit unsigned integer}]
  [request={NOREQ | GENREQ | QUIET}]
  smode={OR0 | OL0 | OR1 | OL1 | ORZ | OLZ | IRM | ILL | IRZ | ILZ}
  [control={OFF | ON}]
  [prv={OFF | ON}]
  [cond_addr={label | 9-bit unsigned integer}]
  cond={UNC | FALL | RISE}
  pin = {pin number}
  [reg={A | B | R | S | T | NONE}]
  [irq={OFF | ON}]
  [data={25-bit unsigned integer}]
}

```

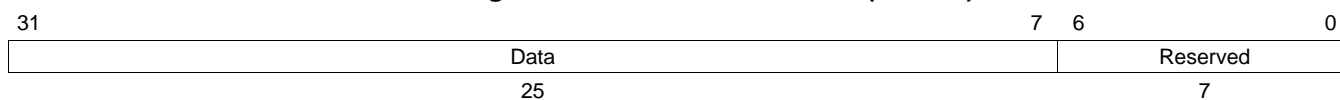
**Figure 19-143. SHFT Program Field (P31:P0)**



**Figure 19-144. SHFT Control Field (C31:C0)**



**Figure 19-145. SHFT Data Field (D31:D0)**



**Cycles** One

**Register modified** Selected register (A, B, R, S or T)

**Description** This instruction shifts the data field of the Instruction. N2HET pins can be used for data in or data out. SHFT includes parameters to select the shift direction (in, out, left, right), shift condition (shift on a defined clock edge on HET[0] or shift always), register for data storage (A, B, R, S or T), and the data pin.

**smode** Shift mode

**Table 19-69. SHIFT MODE Encoding Format**

smode	P3	P2	P1	P0	Operation	
OR0	0	0	0	0	Shift Out / Right	LSB 1st on HETx / 0 into MSB
OL0	0	0	0	1	Shift Out / Left	MSB 1st on HETx / 0 into LSB
OR1	0	0	1	0	Shift Out / Right	LSB 1st on HETx / 1 into MSB
OL1	0	0	1	1	Shift Out / Left	MSB 1st on HETx / 1 into LSB
ORZ	0	1	0	0	Shift Out / Right	LSB 1st on HETx / Z into MSB
OLZ	0	1	0	1	Shift Out / Left	MSB 1st on HETx / Z into LSB
IRM	1	0	0	0	Shift In / Right	HETx into MSB
ILL	1	0	0	1	Shift In / Left	HETx into LSB
IRZ	1	0	1	0	Shift In / Right	HETx in MSB / LSB into Z
ILZ	1	0	1	1	Shift In / Left	HETx in LSB / MSB into Z

**cond** Specifies the shift condition.

**Table 19-70. SHIFT Condition Encoding**

C6	C5	Shift condition
0	X	Always
1	0	Rising edge of HET[0]
1	1	Falling edge of HET[0]

**irq** ON generates an interrupt if the Z flag is set. A value of OFF does not generate an interrupt.  
Default: OFF.

**data** Specifies the 25-bit value for the data field.

### Execution

```

If (SHIFT condition == 0X)
OR (SHIFT condition == 10 AND HET[0] rising edge)
OR (SHIFT condition == 11 AND HET[0] falling edge)
{
  If ([P3:P2] == 00)
  {
    If ((Immediate Data Field == all 0's AND [P3:P0] == 000X)
        OR (Immediate Data Field == all 1's AND [P3:P0] == 001X))
    {
      Z = 1;
    }
    else
    {
      Z = 0;
    }
  }
}

```

```

    }
    else If ([P3:P0] == 1010)
    {
        Z = LSB of the Immediate Data Field;
    }
    else if ([P3:P0] == 1011)
    {
        Z = MSB of the Immediate Data Field;
    }
}

If(((Immediate Data Field == all 0's) AND (Interrupt Enable == 1)) 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 = HET[0] Pin level; (Always executed)

Shift Immediate Data Field once according to P[3:0];

Immediate Data Field = Result of the shift;

Selected register = Result of the shift;

Jump to Next Program Address;

```

---

**NOTE:** The immediate data field evaluates all 0s or all 1s and is performed before the shift operation.

---

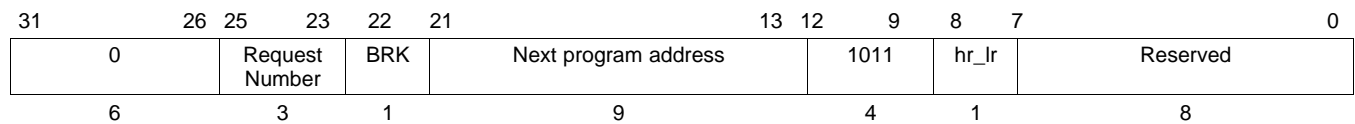
**19.5.3.23 WCAP (Software Capture Word)**

**Syntax**

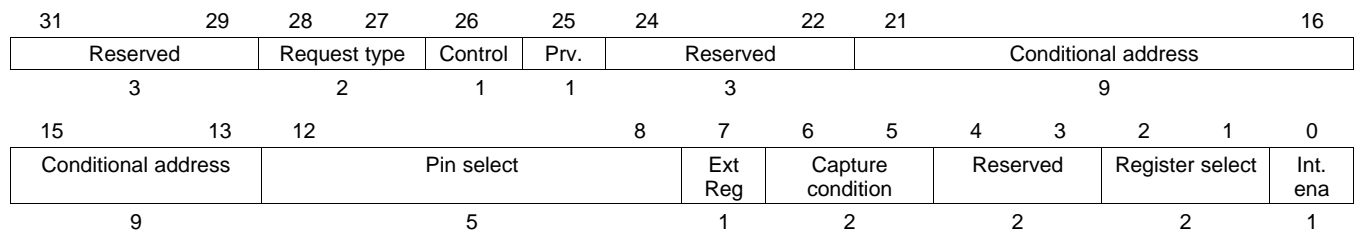
```

WCAP {
  [brk={OFF | ON}]
  [next={label | 9-bit unsigned integer}]
  [reqnum={3-bit unsigned integer}]
  [request={NOREQ | GENREQ | QUIET}]
  [hr_lr={HIGH | LOW}]
  [control={OFF | ON}]
  [prv={OFF | ON}]
  [cond_addr={label | 9-bit unsigned integer}]
  pin = {pin number}
  event={NOCOND | FALL | RISE | BOTH}
  reg={A | B | R | S | T | NONE}
  [irq={OFF | ON}]
  [data={25-bit unsigned integer}]
  [hr_data={7-bit unsigned integer}]
}
    
```

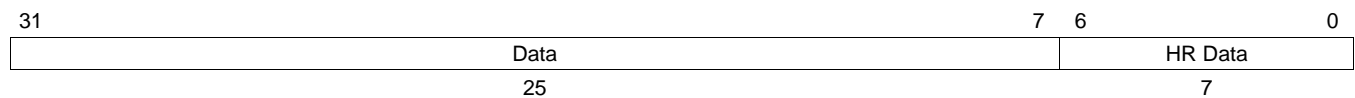
**Figure 19-146. WCAP Program Field (P31:P0)**



**Figure 19-147. WCAP Control Field (C31:C0)**



**Figure 19-148. WCAP Data Field (D31:D0)**



**Cycles** One

**Register modified** None

**Description** This instruction captures the selected register into the data field if the specified capture condition is true on the selected pin. This instruction can be used with all pins.

If the hr\_lr bit is reset, the WCAP instruction will capture an HR time stamp into the data field on the selected edge condition. If the hr\_lr bit is set, the HR capture is ignored.

**event** Specifies the event that triggers the capture.

**Table 19-71. Event Encoding Format for WCAP**

C6	C5	Capture Condition
0	0	Always
0	1	Capture on falling edge
1	0	Capture on rising edge
1	1	Capture on rising and falling edge

**irq** ON generates an interrupt when the capture condition is met. No interrupt is generated for OFF.  
Default: OFF.

**data** Specifies the 25-bit integer value to be written to the data field or selected register.

**hr\_data** HR capture value.  
Default: 0.

**NOTE:** WCAP in HR Mode: The HR Counter starts on a WCAP instruction execution (in the first loop clock) and will synchronize to the next loop clock. When N2HET is turned on and a capture edge occurs in the first loop clock (where the HR counter hasn't been synchronized to the loop clock), then the captured HR counter value is wrong and is of no use. So the captured HR data in the first loop clock should be ignored.

## Execution

```

If (Specified Capture Condition is true on Selected Pin
OR Unconditional capture is selected)
{
    Immediate Data Field = Selected register value;

    If (hr_lr bit == 0) Capture the HR value in Immediate HR Data Field;

    If (Interrupt Enable == 1) SW interrupt flag = 1;
    If ([C28:C27] == 01) Generate request on request line [P25:P23];
    If ([C28:C27] == 11) Generate quiet request on request line [P25:P23];

    Jump to Conditional Address;
}

Jump to Next Program Address;

Prv bit = Current Logic (Lx) value of selected pin; (always executed)

```

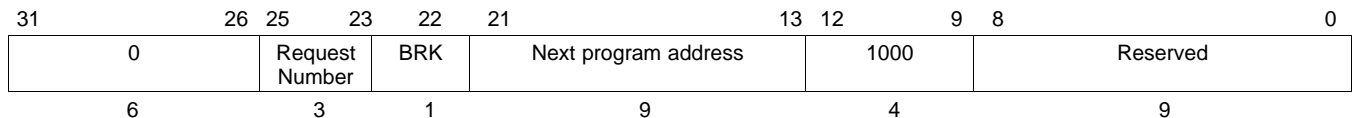
**19.5.3.24 WCAPE (Software Capture Word and Event Count)**

**Syntax**

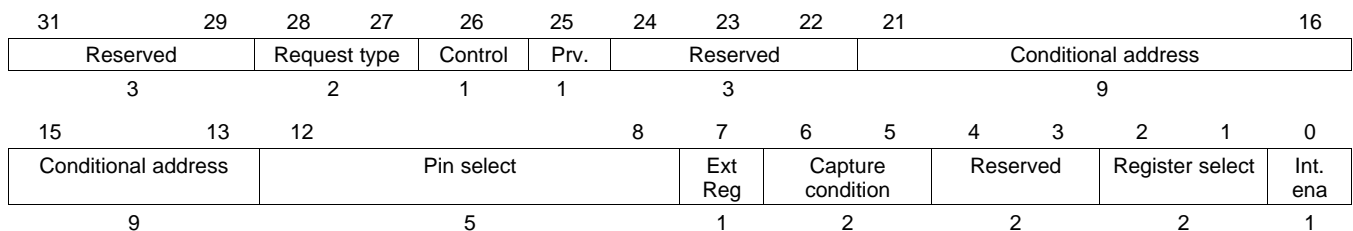
```

WCAPE {
  [brk={OFF | ON}]
  [next={label | 9-bit unsigned integer}]
  [reqnum={3-bit unsigned integer}]
  [request={NOREQ | GENREQ | QUIET}]
  [control={OFF | ON}]
  [prv={OFF | ON}]
  [cond_addr={label | 9-bit unsigned integer}]
  pin = {pin number}
  event={NOCOND | FALL | RISE | BOTH}
  [reg={A | B | R | S | T | NONE}]
  [irq={OFF | ON}]
  [ts_data={25-bit unsigned integer}]
  [ec_data={7-bit unsigned integer}]
}
    
```

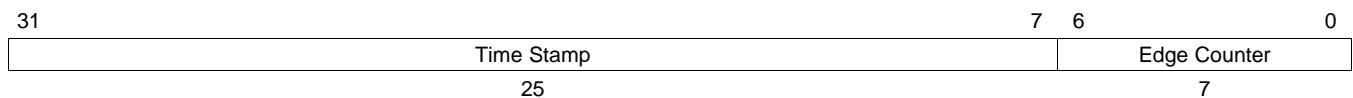
**Figure 19-149. WCAPE Program Field (P31:P0)**



**Figure 19-150. WCAPE Control Field (C31:C0)**



**Figure 19-151. WCAPE Data Field (D31:D0)**



**Cycles** One

**Register modified** None

**Description** This instruction captures the selected register into the data field [D31:D7] and increments an event counter [D6:D0] if the specified capture condition is true on the selected pin. This instruction can be used with all pins, but the time stamp [D31:D7] has loop resolution only.



**event** Specifies the event that triggers the capture.

**Table 19-72. Event Encoding Format for WCAPE**

C6	C5	Capture Condition
0	0	Always
0	1	Capture on falling edge
1	0	Capture on rising edge
1	1	Capture on rising and falling edge

**irq** ON generates an interrupt when the capture condition is met. No interrupt is generated for OFF.  
Default: OFF.

**ts\_data** Specifies the 25-bit integer value for [D31:D7]  
Default: 0.

**ec\_data** Specifies the initial 7-bit integer value for [D6:D0].  
Default: 0.

## Execution

```

If (Specified Capture Condition is true on Selected Pin
OR Unconditional capture is selected)
{
    Immediate Data Field[31:7] = Selected register value;
    Immediate Data Field [6:0] = Immediate Data Field [6:0] + 1;

    If (Interrupt Enable == 1) SW interrupt flag = 1;
    If ([C28:C27] == 01) Generate request on request line [P25:P23];
    If ([C28:C27] == 11) Generate quiet request on request line [P25:P23];

    Jump to Conditional Address;
}

Jump to Next Program Address;

Prv bit = Current Logic (Lx) value of selected pin; (always executed)

```

## High End Timer Transfer Unit (HTU) Module

This chapter describes the high-end timer transfer unit (HTU) module. The HTU is similar to the DMA (Direct Memory Access) module, but it is specialized to transfer NHET (High End Timer) data to or from the microcontroller RAM.

Topic	Page
<b>20.1 Overview</b> .....	<b>1048</b>
<b>20.2 Module Operation</b> .....	<b>1049</b>
<b>20.3 Use Cases</b> .....	<b>1062</b>
<b>20.4 Control Registers</b> .....	<b>1065</b>
<b>20.5 Double Control Packet Configuration Memory</b> .....	<b>1098</b>
<b>20.6 Examples</b> .....	<b>1107</b>

## 20.1 Overview

The HET transfer unit is a dedicated direct memory access controller which transfers data between the NHET RAM and RAM buffers located in the main memory address range. This eliminates time consuming CPU accesses to the NHET RAM to gather measurement data or creating output waveforms and thus freeing up the CPU to perform other tasks.

### 20.1.1 Features

- Independently transfers data between the NHET and the main memory
- 8 double control packets supporting dual buffer configuration
- Transfer requests generated by NHET instructions/events
- One shot, circular and auto switch buffer transfer modes for each double control packet for flexible buffer handling
- Constant and post-increment addressing modes
- 32- or 64-bit transactions
- Programmable memory protection region
- Parity protect control packet RAM
- Extensive diagnostic functionality

## 20.2 Module Operation

The HTU is tightly coupled to the NHET and is not intended to transfer data from other peripheral modules. It initiates transfers with the help of requests generated by the NHET program and configurable control packets. Figure 20-1 shows a system block diagram of the HTU and the main path for the data transfer. The tight coupling and the dedicated bus into the SCR (Switched Central Resource) reduces the amount of data transferred on the peripheral bus, which increases the overall system performance. However if the application decides to use the direct CPU access method to the NHET RAM, it is free to do so.

Figure 20-1. System Block Diagram

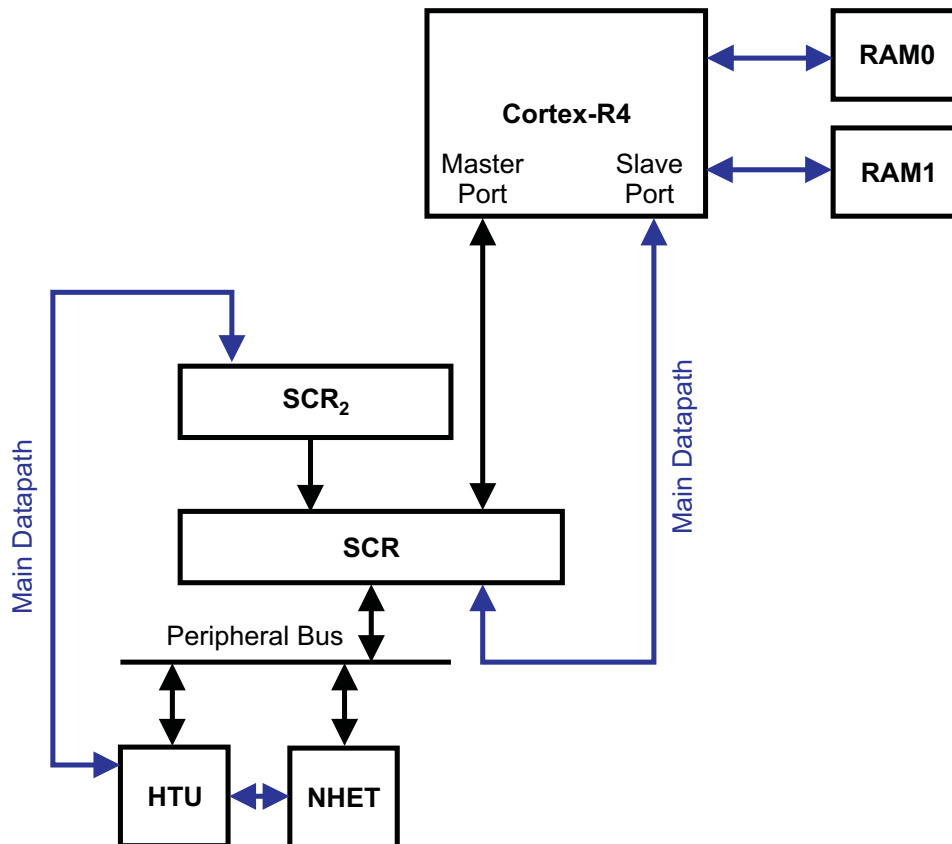
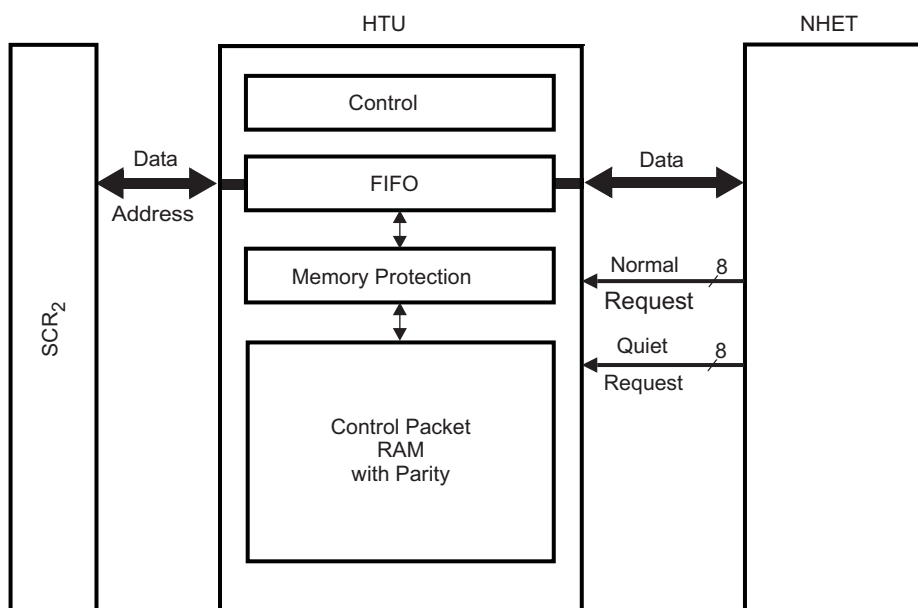


Figure 20-2 shows a more detailed block diagram of the HTU module.

**Figure 20-2. HTU Block Diagram**

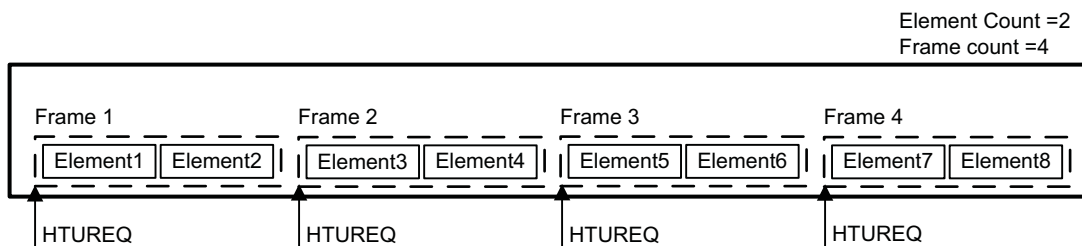
Transfers between NHET RAM and the main memory are triggered by 8 different normal NHET requests. Quiet requests are used for specific cases and are discussed in [Section 20.2.4.1](#). Control packets, which store the source and destination addresses, the transfer count and other information (see [Section 20.5](#)), are associated with the requests. A FIFO decouples the read- and write-path and allows to do data-packing in the case of different read- and write-data sizes. The application can specify a section of memory into or from which the data is transferred. This serves as memory protection in the case that information in the control packet RAM was unintentionally altered and avoids that the HTU can overwrite important application data.

Control packets are implemented as double control packets (DCP) which allow to specify two buffers for the data transfer. This enables the CPU to work with one buffer, while new data is transferred to/from the other buffer.

The control packet defines:

- the start address of the source/destination buffers
- the NHET instruction address location
- how many elements need to be transferred
- the buffer handling

A transfer is triggered when a certain condition (e.g. capture, compare condition) is detected by a NHET instruction. The NHET instruction specifies which request line to the HTU will be triggered at the event. The DCPs have a fixed assignment to the request lines and the corresponding assignment can be found in the device datasheet. Once a request is triggered, it starts a frame transfer. A frame can contain one or more elements. Elements are defined as 32-bit or 64-bit words of data.

**Figure 20-3. Example of a HTU Transfer**

## **20.2.1 Data Transfers between Main RAM and NHET RAM**

### **20.2.1.1 Addressing Modes**

The addressing modes of a control packet need to be distinguished between the main RAM of the CPU and the NHET RAM.

#### **Main RAM**

For each double control packet (see [Section 20.2.1.3](#)) the addressing mode for the main RAM (RAM0/1) can be configured to constant or post-increment mode in register IHADDRCT.

Constant Addressing	In constant mode, the HTU writes/reads the data to/from the same address in the main RAM.
Post-increment Addressing	In post-increment mode, the HTU writes/reads the data to/from the main RAM by incrementing through the addresses after each transfer. If 32-bit transfers are selected it will automatically increment by 4 Byte, if 64-bit transfers are selected, it will increment by 8 Byte. The examples of Use Cases illustrate the post-increment mode, where the elements of consecutive frames are transferred to/from consecutive locations in the main RAM buffer.

#### **NHET RAM**

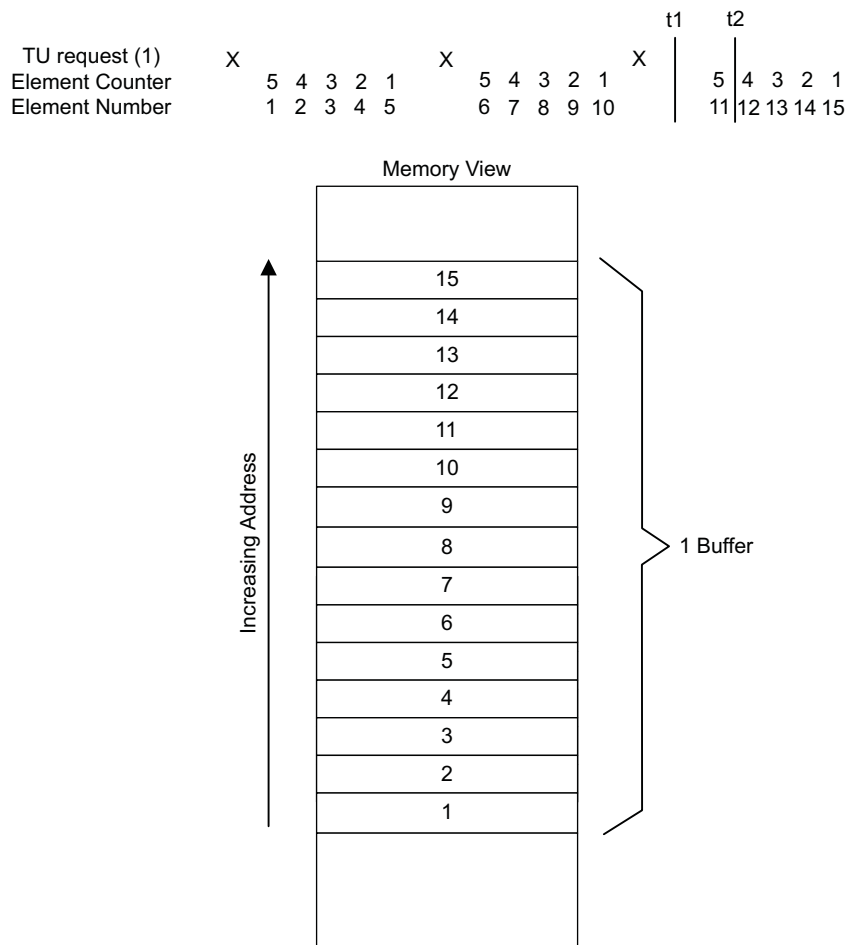
How a DCP addresses the NHET RAM is determined by the initial NHET address, the initial element counter (IETCOUNT) and the NHET addressing mode (ADDMH). The main difference to the main RAM addressing mode is that the HET address is reset to the initial HET address for every first element of a frame. To implement constant addressing, the initial element counter needs to be set to 1. Post-increment addressing is selected by programming the initial element counter to a value other than 1.

### **20.2.1.2 Single Buffer Implementation**

In a single buffer implementation, the DCP is set up to transfer data to/from a single buffer in the main RAM. With each transfer request, the programmed number of elements is transferred and the buffer pointer is reset to its starting address after the programmed number of frame transfers have completed.

[Figure 20-4](#) shows the request on one request line of the HTU and the frame running on the assigned control packet visualized by the element counter. In the diagram the frame has 5 element transfers (i.e. element count = 5).

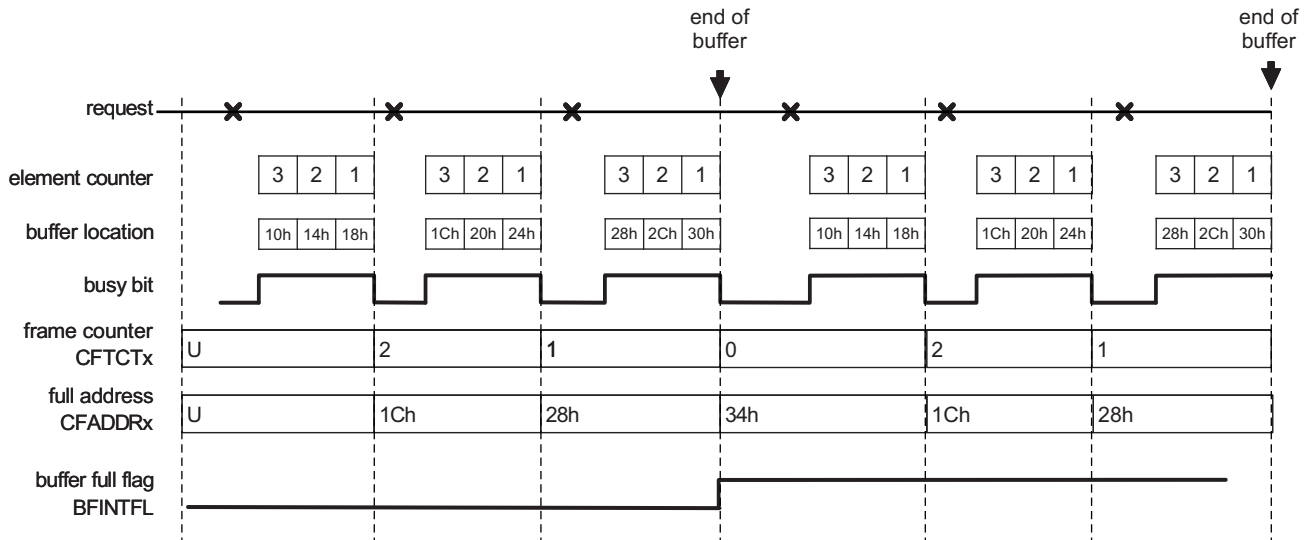
**Figure 20-4. Single Buffer Timing and Memory Representation**



Before the application reads the buffer, it has to disable the control packet to avoid that new data overwrites the buffer while it's being accessed by the application. Regardless of the control packet being disabled at t1 or t2 the last frame will always be completed, since the trigger request has been received already. The application can determine any ongoing transfers by the TIPF flag and the NACP bits.

- One Shot Buffer Mode**      If TMBA or TMBB is set to one shot buffer mode then the data stream will stop after all elements of buffer A or buffer B have been transferred. This means that the corresponding DCP will be disabled after the last frame was transferred to/from buffer A or B and CFTCTA or CFTCTB decrements to zero.
- Circular Buffer Mode**      If TMBA or TMBB is set to circular buffer mode, then the data stream will continue back at the start of buffer A or B after all elements of buffer A or B have been transferred. The example of Timing Example for Circular Buffer Mode assumes IETCOUNT=3 (Initial Element Transfer Count), IFTCOUNT=3 (Initial Frame Transfer Count, SIZE=0 (Size of Transfer = 32-bit) and ADDFM=0 (Addressing Mode Main Memory = Post Increment). So there are in total 9 32-bit values in the buffer. It also assumes IFADDRx=10h. "U" means uninitialized.

Figure 20-5. Timing Example for Circular Buffer Mode



### 20.2.1.3 Dual Buffer Implementation

The transfer unit provides **double control packets (DCPs)** supporting the use of two buffers per data stream (i.e. per HTU request source). If one buffer should be read by the CPU or DMA, the data stream is directed to the other buffer and the first buffer is frozen. Switching to the other buffer can be triggered with a write access to the CPENA register or with the DCP configured to automatically switch to the other buffer when the programmed number of frames has been transmitted. Freezing the buffer avoids this buffer to be overwritten with new HET data while the CPU or DMA reads this buffer.

Figure 20-6 shows a timing example of two HET instructions 1 and 2, which are the request sources for the HTU (and are controlled by DCP 1 and DCP 2). Each generated frame has 5 element transfers. Request source 1 has two RAM buffers, controlled by two control packets 1A and 1B. Request source 2 has two RAM buffers, controlled by two control packets 2A and 2B.



**Figure 20-6. Dual Buffer Timing**

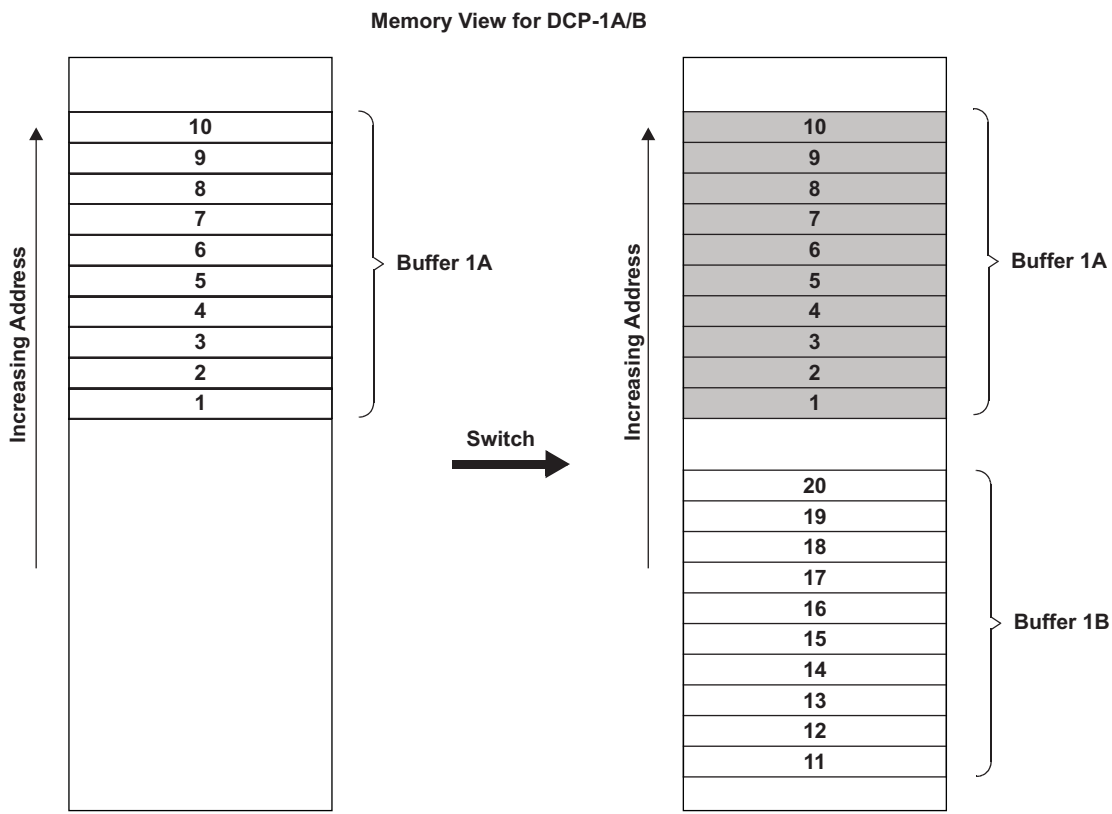
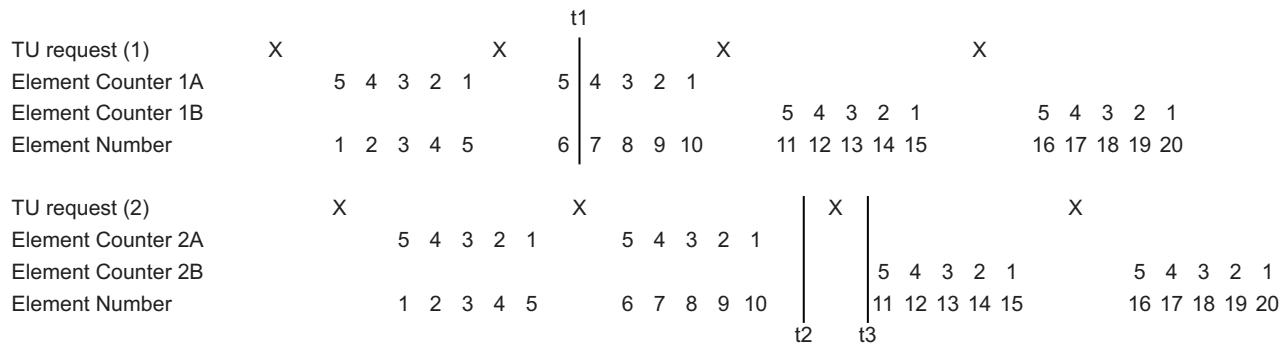


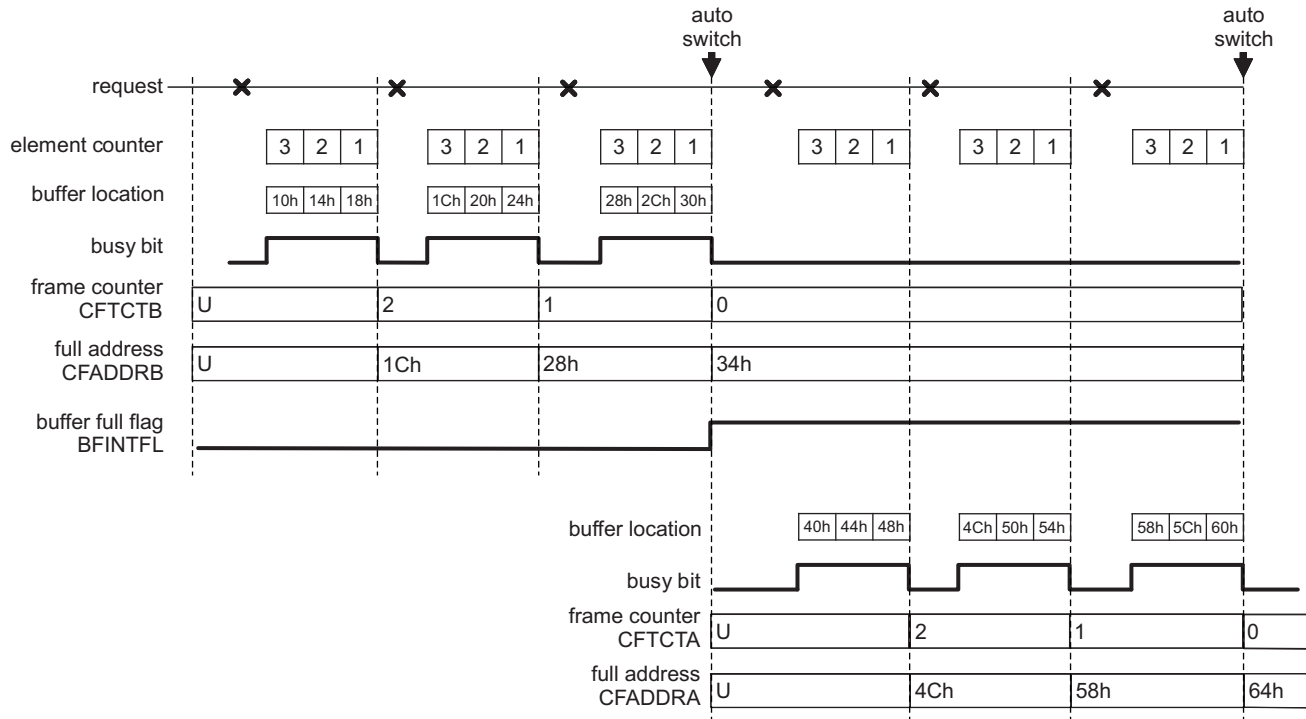
Figure 20-6 shows a switch at time t1, where buffer 1A is frozen and data stream 1 is directed to buffer 1B, but only after the frame has been completed. It also shows the time (t2 or t3) where 2A is frozen and data stream 2 is directed to buffer 2B. If the switch happens between the request and the start of the frame (e.g. time t3), then the frame is processed by the new control packet (although the old control packet was active at the time of the request). The delays between the HTU requests and the start of the element transfers result from the fact that the HTU can process only one transfer at a time.

**Auto Switch Buffer Mode**

If TMBA is set to auto switch mode, then the data stream will continue at the start of buffer B after all elements of buffer A have been transferred. This means that in the CPENA register, CP A is disabled and CP B is enabled automatically and buffer B uses its initial main memory address and initial frame counter to start. The same principle is valid for TMBB and buffer B.

The examples of Figure 20-7 assumes IETCOUNT=3 (Initial Element Transfer Count), IFTCOUNT=3 (Initial Frame Transfer Count, SIZE=0 (Size of Transfer = 32-bit) and ADDFM=0 (Addressing Mode Main Memory = Post Increment). So there are in total 9 32-bit values in buffer A and B. It also assumes IFADDRB=10h and IFADDRA=40h. "U" means uninitialized.

Figure 20-7. Timing Example for Auto Switch Buffer Mode



20.2.1.4 General Control Packet Behavior

The action defined by the selected mode will be performed at the end of the last frame, which has the frame counter value of 1. The one shot and auto switch mode will automatically update the CPENA register at this time. Note, that for all three modes listed above, it is possible to switch to the other buffer by writing to CPENA before the end of the current buffer is reached.

If a write access to CPENA happens while the last frame of DCP x (with frame counter = 1) is transferred then the priority is defined by Table 20-1.

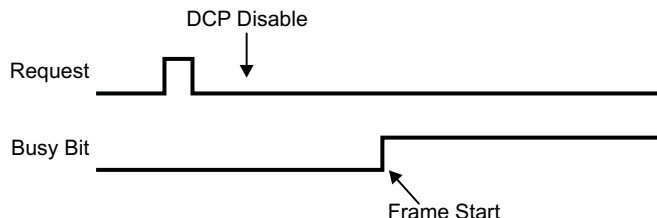
Table 20-1. CPENA / TMBx Priority Rules

Write access to CPENA bits (2-x+1) and (2-x) during the frame with frame counter = 1 ( <sup>(1)</sup> )	Priority Rule
Disable: 01 --> 00 or 10 --> 00	Disabling the DCP by the write to CPENA has priority, TMBx is ignored.
Stay: 01 --> 01 or 10 --> 10	The write access to CPENA is ignored, TMBx has priority and defines the action.
Switch: 01 --> 10 or 10 --> 01	Switching the DCP by the write to CPENA has priority, TMBx is ignored.

(1) See read table of CPENA register

There could be a case where the CPU wants to do main memory operations, but does not want the HTU modifying the main memory. It could happen that a request was already active, but the frame transfer hasn't started yet when the application disabled the control packets. The timing diagram in [Figure 20-8](#) shows this scenario.

**Figure 20-8. Timing for Disabling Control Packets**



Since the request for the transfer was already received before the DCPx is disabled, the HTU will still start the frame transfer. The application would poll the BUSYx bit during the time the DCPx was disabled and before the frame was started and would read a non-busy information. It then would start the main memory operations thinking all transfers have completed, however after some time the HTU will start the outstanding frame transfer and corrupt the main memory.

To avoid this, the application can set the VBUSHOLD bit to disable all transactions between the HTU and the main memory. It has to poll the BUSBUSY bit to ensure that no outstanding transactions on the bus are pending. The HTU will still receive all transfer requests from the NHET, but it will not be able to transfer any data to or from the main memory, while the VBUSHOLD bit is set.

### 20.2.2 Arbitration of HTU Elements and Frames

- Frames do not interrupt each other. If a request occurs on DCP x while another frame runs on DCP y (and  $x \neq y$ ), then the current frame completes before the new frame starts.
- If two or more request lines are active, the request line with the lower number (specified in the request number field of the corresponding NHET instruction) is serviced first.

### 20.2.3 Conditions for Frame Transfer Interruption

If a frame is currently transferred on DCP x and one of the events listed below happens, then the event will (1.) clear the element counter of DCP x, (2.) stop new element transfers on DCP x (3.) clear the active busy bit of DCP x and (4.) disable DCP x in the CPENA register. The DCPs other than DCP x will not be affected.

- Request Lost Error of DCP x (with CORL bit set to zero).
- Parity Error of DCP x (with parity check enabled and COPE bit set to zero). See also Control Packet RAM Parity Checking
- Bus Error of DCP x.
- Memory Protection Error of DCP x (with memory protection enabled). See also [Section 20.2.5](#)
- Writing a one to a BUSY bit (belonging to DCP x) if that bit is one. There is no effect if the BUSY bit is zero.
- Writing a one to the HTURES bit.

When a memory protection error occurs, the access to the protected address is blocked. The frame is stopped before the element, which caused the violation transfer, starts. All other errors will let the current element transfer finish.

In case of the Request Lost and Bus Error one more element transfer goes on the bus, before the frame is actually stopped. Accordingly the busy bit is cleared after the element, which follows the element that caused the error.

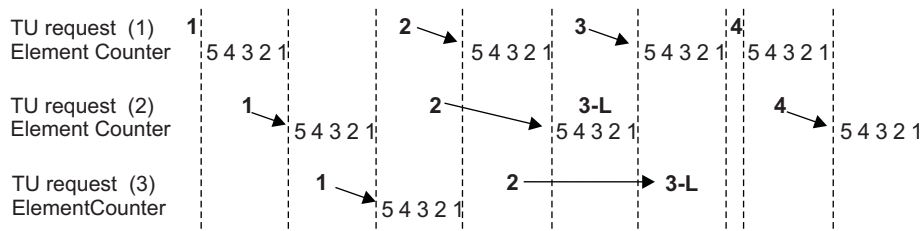
In case of the Bus Error, the counter for the element, which follows the element that caused the error, is captured to the ERRETC register field.

**NOTE:** If the HTUEN bit is cleared during a frame is transferred, then the frame will be completed before the HTU is disabled.

### 20.2.4 HTU Overload and Request Lost Detection

If the number of different HTU request sources is "high", the period between the requests is "short" and/or the initial element counter values are "big", then the HTU could get into a overload situation. In Timing Example Including Lost Requests all requests marked with "L" are lost, since their frame is not completed at the time the next request occurs. Each number in the rows "TU request (x)" represents a time, where the associated NHET instruction generates a request on DCP x. The arrows in Figure 20-9 point to the associated frame, which could be delayed compared to the request. The delays are caused by different frames, which are currently processed. Figure 20-9 assumes that the CORL bit in the RLBECTRL register is set, which causes the DCP to stay enabled and let the data stream continue after a request lost error occurred on the DCP (see 3-L for TU request (2)).

**Figure 20-9. Timing Example Including Lost Requests**



Lost requests are signaled with the RLOSTFL register and if enabled can generate request lost interrupts.

If the CORL bit is set, a frame will be completed and the corresponding DCP stays enabled even if a request lost was generated during this frame.

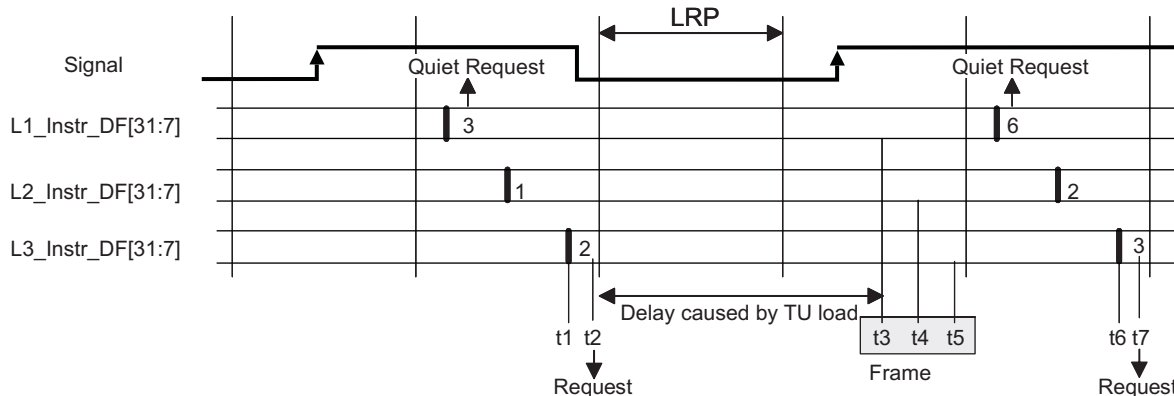
In dual buffer mode the request lost detection works continuously, independent of the CP switches.

#### 20.2.4.1 Requests and Quiet Requests

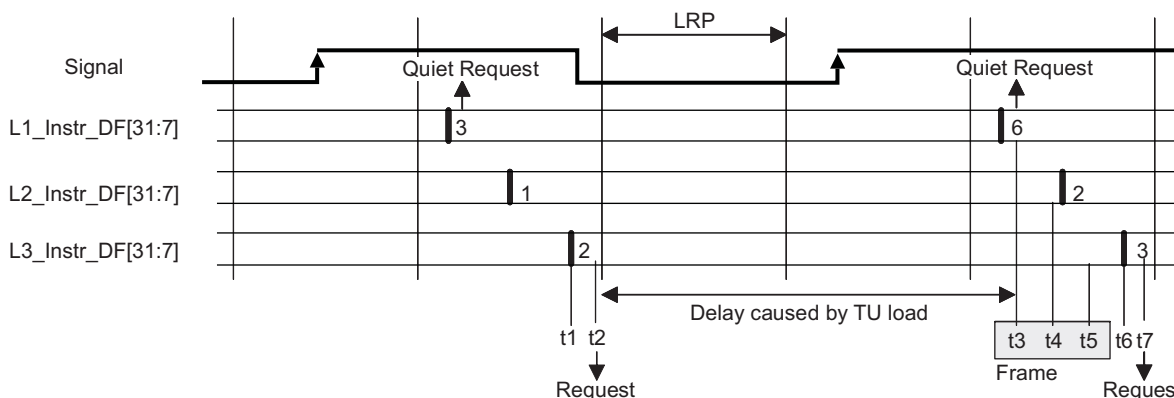
In addition to generating too many transfer requests and thus overloading the HTU and not being able to transfer data at all, it can happen that inconsistent data is transferred. The following examples illustrate such scenarios.

In the examples below the HTU reads a frame of three elements from the datafield of three different instructions. In Figure 20-10 the L3-Instruction generates the HTU request at time t2, t7, etc. and the according frame (at t3). The frame is delayed because of the HTU load. However, as shown in Figure 20-10, the delay still allows the frame to complete before the datafield of instruction L1 is updated again. However when the delay is longer (as shown in Figure 20-11) then the frame could fall into the NHET loop (LRP), in which the NHET updates the data fields of the L1, L2 and L3 instructions. In this case the HTU could read inconsistent data as shown in the diagram: A wrong (new) value is read from L1 (at time t3), but correct ("old") values are read from L2 and L3 (at times t4 and t5).

**Figure 20-10. Timing, which Generates No Request Lost Error**



**Figure 20-11. Timing, which Generates a Request Lost Error**



To ensure consistent data the NHET instructions are able to generate a **quiet request**, which does not originate a transfer, but is only used by the HTU for consistency check. If a frame has not completed since the last request (or not even started) at the time the quiet request occurs, then the HTU signals a request lost error. All instructions, which allow to generate a request can be configured to generate a quiet request instead. So in the examples of [Figure 20-10](#) and [Figure 20-11](#) instruction L1 should be configured to generate a quiet request and instruction L3 to generate a normal request. In the case of [Figure 20-11](#) the corresponding bit in the RLOSTFL register will be set.

It is the responsibility of the NHET software to enable a quiet request for the first instruction of an instruction block, which is addressed by DCP x, and to enable a normal request only for the last instruction of this block. Since enabling the quiet request should enable a proper request lost detection for DCP x, both NHET instructions need to specify the same DCP x (reqnum=x).

The control fields of the HET instructions provide a 2-bit field to configure one of the following possibilities (as shown in [Table 20-2](#)). A 3-bit field in the program field will select which of the 8 Double Control Packets will be triggered by the request.

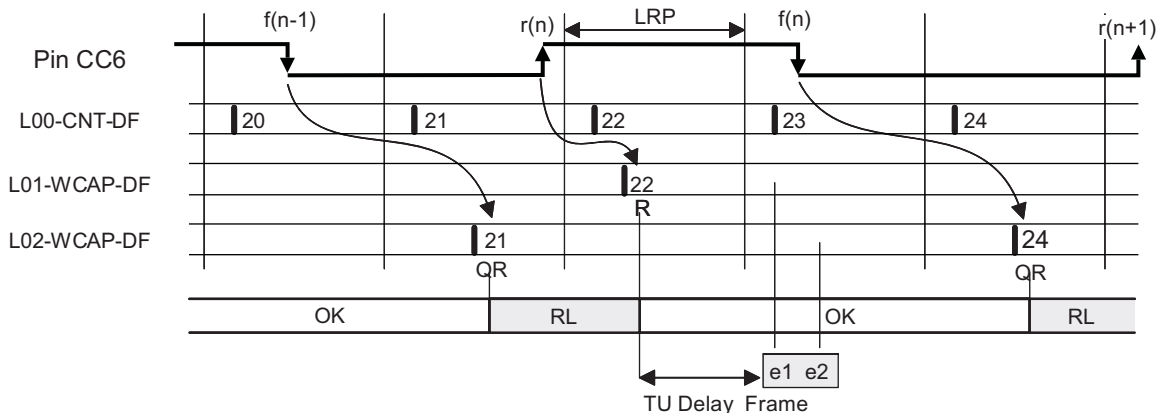
**Table 20-2. Triggered Control Packets**

Request Type Bit 1	Request Type Bit 0		Request Number
Don't care	0	No request	Specify number 0, 1,... or 7, which selects the HTU or DMA request line
0	1	Generate normal request	
1	1	Generate quiet request	

In the case of very light HTU load, but higher signal requirements (e.g. high frequency), the quiet request could also be used to define periods in which the data read by a control packet is safe. The following HET code will capture counter time stamps to the L1-WCAP data field after rising edges (at pin CC6) and to the L2-WCAP data field after falling edges (at pin CC6):

```
L0 CNT {reg=A, max=0x1FFFFFFF}
L1 WCAP {reqnum=3, request=GENREQ, event=RISE, reg=A, pin=CC6}
L2 WCAP {reqnum=3, request=QUIET, event=FALL, reg=A, pin=CC7}
; HET HRSHARE feature configured to assign both WCAPs to pin CC6
```

Figure 20-12. Timing Example for Two WCAP Instructions



The HTU frame will have two elements: The first gives the time stamp of the rising edge  $r(n)$  and the second the one of the previous falling edge  $f(n-1)$ . Using the code above requests (R) and the quiet requests (QR) will occur at the times shown in Figure 20-12 and a request lost will only be signaled when the frame makes an access during the times marked with RL. So reading [22, 21] as frame elements is correct. If the signal frequency would increase, then a wrong pair [22,24] could be read, but this will be signaled by a request lost error, since at least  $e2$  falls into the RL period.

### 20.2.5 Memory Protection

This feature allows restricting accesses to certain areas in memory in order to protect critical application data from unintentionally being manipulated by the HTU.

If the HTU memory protection feature is disabled the full 4 GB address range can be accessed by the HTU without exception. There are two memory regions which start and end addresses can be configured. With the memory protection feature enabled, read and write accesses by the HTU inside the defined regions are allowed. For accesses outside the regions, one of two modes is configurable:

- Any access performed by the HTU is forbidden and will be signaled to the ESM module. Write accesses will be blocked.
- Read access is allowed but write access will be blocked and signaled to the ESM module.

To use one region only, REG01ENA must be zero. Bits ACCR01, INTENA01, and register settings of MP1S, and MP1E will be ignored.

To use both regions below rules must followed:

1. Memory mapped region 0 covers a lower memory area as Memory mapped region 1.
2. REG01ENA is a one and REG0ENA is a zero.
3. ACCR01 is set for the desired access type, ACCR0 is ignored.
4. INTENA01 is set for the desired action, INTENA0 is ignored.

If an element transfer of DCP x generates a memory protection error, then:

1. The element counter of DCP x is cleared
2. All new element transfers on DCP x are stopped
3. The active busy bit of DCP x is cleared

4. DCP x is disabled in the CPENA register. The DCPs other than DCP x will not be affected.
5. The FT flag will be set
6. An error is signaled to the ESM module

### 20.2.6 Control Packet RAM Parity Checking

The HTU module can detect parity errors in the DCP (Double Control Packet) RAM. DCP RAM parity checking is implemented using one parity bit per byte. Even or odd parity checking can be selected in the DEVCR1 register of the system module and can be enabled/disabled by a 4-bit key in the PCR register.

During a read access to the DCP RAM, the parity is calculated based on the data read from the RAM and compared with the good parity value stored in the parity bits. The parity check is performed when the HTU or any other master (e.g. CPU) makes a read access to the DCP RAM. A read access within the RAM section of an initial or current DCP checks all 16 bytes of the DCP at a time (see also DCP memory map). For example, if a byte read access happens for DCP RAM address 0, but there is a parity error at byte address Ch then the parity error will occur and the captured parity address will be Ch and not 0. The address of the byte in which the error occurred can be read from the PAR register. If successive DCP RAM read accesses generate multiple parity errors, only the address of the first detected error will be captured and the PAR register will not be updated by subsequent errors until it is read by the application. When multiple errors in a 16 byte word are detected, only the address of the lowest byte will be captured.

The application can decide whether to stop any transfers when a parity error is detected or to continue transferring data. If the COPE (Continue On Parity Error) bit is zero and parity checking is enabled, then the HTU will not start the frame and the corresponding DCP will be automatically disabled in the CPENA register. If a master other than the HTU (e.g. CPU) reads the RAM section of DCP x and a parity error is detected during this read access, while the parity check is enabled and the COPE bit is zero, then the DCP x will be automatically disabled in the CPENA register. If a frame for this DCP x is ongoing during this read access, then in addition the element counter of DCP x is cleared, all new element transfers on DCP x are stopped and the active busy bit of DCP x is cleared. With COPE set to one and the parity check enabled, the parity checking will still be performed, but the data transfer of an active DCP continues after a parity error was detected for this DCP. So neither the DCP with the parity error will be disabled nor the frame will be stopped.

After a DCP is enabled (with CPENA using BIM=0) then at the start of the first frame the HTU performs the parity check only on the initial DCP, since it does not need the current DCP information. For further frames the HTU performs the parity check for both initial and current DCP, since it needs both information.

On a parity error detection an error will also be signaled to the ESM module.

#### 20.2.6.1 Parity Bit Mapping and Testing

To test the parity checking mechanism, the parity RAM can be made accessible in order to allow manual fault insertion. Once the TEST bit is set, the parity bits are mapped to address FF4E 0200h.

When in test mode (i.e. the parity RAM is accessible) no parity checking will be done when reading from parity RAM, but parity checking will still be performed for read accesses to the DCP RAM.

[Table 20-3](#) and [Table 20-4](#) show how the corresponding parity bits of the DCP RAM bytes are mapped into the memory.

**Table 20-3. DCP RAM**

Bit	31	24	23	16	15	8	7	0
FF4E 0000h	Byte 0		Byte 1		Byte 2		Byte 3	
FF4E 0004h	Byte 4		Byte 5		Byte 6		Byte 7	
FF4E 0008h	Byte 8		Byte 9		Byte 10		Byte 11	
FF4E 000Ch	Byte 12		Byte 13		Byte 14		Byte 15	

**Table 20-4. DCP Parity RAM**

Bit	24	16	8	0
FF4E 0200h	P0	P1	P2	P3
FF4E 0204h	P4	P5	P6	P7
FF4E 0208h	P8	P9	P10	P11
FF4E 020Ch	P12	P13	P14	P15

Each byte in DCP RAM has its own parity bit in the DCP Parity RAM. P0 is the parity bit for byte 0, P1 is the parity bit for byte 1 and so on.

### 20.2.6.2 Initializing Parity Bits

After device power up, the DCP RAM content including the parity bit cannot be guaranteed. In order to avoid parity failures, when reading DCP RAM, the RAM has to be initialized first. This can simply be done by writing known values into the RAM by software and the corresponding parity bit will be automatically calculated.

Another possibility to initialize the DCP memory and its parity bits is to use the system module, which is an on-chip module external to the HTU. This module can start the automatic initialization of all RAMs on the microcontroller including the HTU DCP RAM. This function initializes the complete DCP RAM to "0" when activated by the system module. Depending on the even/odd parity selection, all parity bits will be calculated accordingly. The HTUEN bit must be cleared and the parity functionality must be enabled (by PARITY\_ENA) during the automatic DCP RAM initialization. If HTUEN is one when the initialization is triggered by the system module, then the initialization will not be performed and the HTU operation is not affected. If a one is written to HTUEN during the initialization, then the HTUEN bit will be set but the HTU will not be enabled before the initialization completes.



## 20.3 Use Cases

### 20.3.1 Example: Single Element Transfer with One Trigger Request

This example considers the case that the HTU fills a RAM buffer in the main (CPU) data RAM. The HTU reads from the instruction which generates the HTU requests.

This example uses a PCNT instruction. Every time the PCNT has captured a new pulse or period value, it will automatically generate a transfer request to the HTU, which then transfers the value from the NHET RAM to the buffer RAM. So over time consecutive locations in the RAM buffer can be filled with consecutive measurement values captured into the NHET RAM data field of the same PCNT instruction without loading or interrupting the CPU.

### 20.3.2 Example: Multiple Element Transfer with One Trigger Request

The following example shows how the HTU could be used to fill a RAM buffer with a data stream including different types of measurement values belonging to the same NHET input signal (on one pin): Time stamp values (WCAP), edge counter values (ECNT) and last period values (PCNT).

Figure 20-13 shows the timing and Table 20-5 shows the byte addresses of the program- (PF), control- (CF), data- (DF) and reserved field (res) of the WCAP-ECNT-PCNT instruction block. The timing and code example assumes that all three instructions are assigned to the same NHET pin.

Figure 20-13. Timing of the WCAP, ECNT, PCNT Example

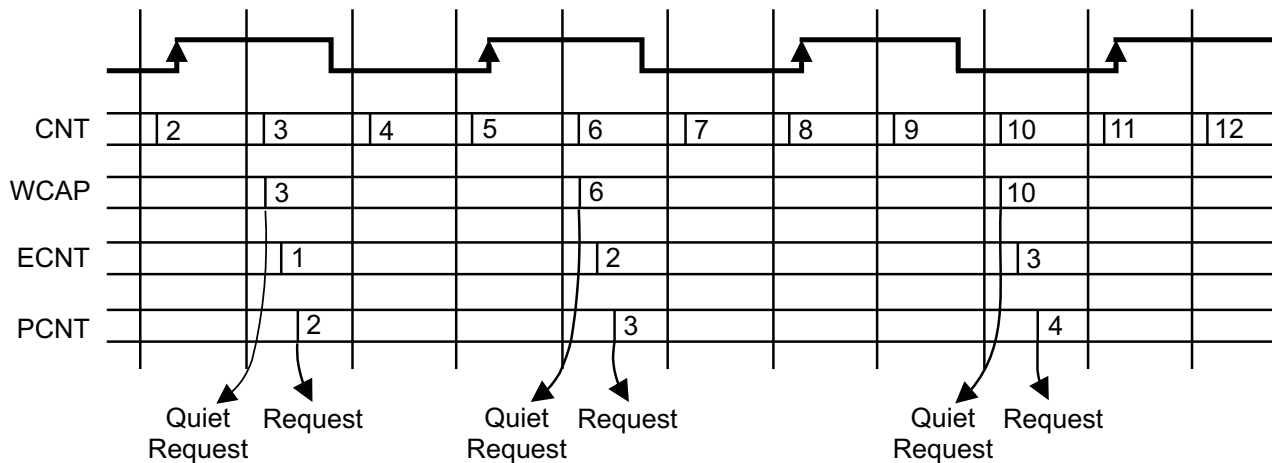


Table 20-5. Field Addresses of the WCAP, ECNT, PCNT Example

	PF	CF	DF	Res
WCAP	30h	34h	38h	3Ch
ECNT	40h	44h	48h	4Ch
PCNT	50h	54h	58h	5Ch

In the HET code the HTU request is enabled only for the last instruction (PCNT) of the WCAP-ECNT-PCNT block. When the PCNT condition is true, it will cause the generated HTU frame to perform three HTU element reads from the data fields of WCAP, ECNT and PCNT.

#### 32-Bit-Transfer of data fields:

The following diagram shows how the internal element counter, frame counter and the address registers change over time for the example described above. Every time the PCNT instruction captures a new value it generates a request to the HTU, which starts a frame. At the end of each frame the frame counter decrements.

**Table 20-6. 32-Bit-Transfer of Data Fields <sup>(1)</sup>**

Frame Counter	3			2			1		
Element Counter	3	2	1	3	2	1	3	2	1
Source Address (HET)	38h	48h	58h	38h	48h	58h	38h	48h	58h
Destination Address (main CPU RAM)	70h	74h	78h	7Ch	80h	84h	88h	8Ch	90h

<sup>(1)</sup> The diagram shows the byte addresses

The destination buffer is filled with the WCAP, ECNT and PCNT data field values as shown below.

**Table 20-7. Destination Buffer Values**

Address	Frame Count	Instruction	Value
70h	3	WCAP	3
74h	3	ECNT	1
78h	3	PCNT	2
7Ch	2	WCAP	6
80h	2	ECNT	2
84h	2	PCNT	3
88h	1	WCAP	10
8Ch	1	ECNT	3
90h	1	PCNT	4

The corresponding setup of the HTU control packet for this example is as follows:

```

IHADDR   = 0x38           // points to WCAP data field
IFADDRA  = 0x70           // points to buffer
ITCOUNT [frame count = 3] [element count = 3]
IHADDRCT = [DIR: Read HET and write to full address]
           [SIZE: 32 bit]
           [ADDMH: Increment HET address by 16 bytes]
           [ADDMF: Post increment full address mode]
           [Any transfer mode]

```

### 20.3.3 Example: 64-Bit-Transfer of Control Field and Data Fields

The following diagram shows how the internal element counter, frame counter and the address registers change over time assuming the same example as in [Section 20.3.2](#), but now with a transfer size set to 64-bit. The HET address now points to the control field of the instruction, so CF and DF are transferred as 64 bit data.

**Table 20-8. 64-Bit-Transfer of Control Field and Data Fields <sup>(1)</sup>**

Frame Counter	3			2			1		
Element Counter	3	2	1	3	2	1	3	2	1
HET (Source) Address	34h	44h	54h	34h	44h	54h	34h	44h	54h
Full (Destination) Address	70h	78h	80h	88h	90h	98h	A0h	A8h	B0h

<sup>(1)</sup> The diagram shows the byte addresses.

The destination buffer is filled with the WCAP, ECNT and PCNT control and data field values as shown on the right.

**Table 20-9. Destination Buffer Values**

Address	Frame Count	Instruction	Value
70h	3	WCAP	Control Field Value
74h	3	WCAP	3
78h	3	ECNT	Control Field Value
7Ch	3	ECNT	1
80h	3	PCNT	Control Field Value
84h	3	PCNT	2
88h	2	WCAP	Control Field Value
8Ch	2	WCAP	6
90h	2	ECNT	Control Field Value
94h	2	ECNT	2
98h	2	PCNT	Control Field Value
9Ch	2	PCNT	3
A0h	1	WCAP	Control Field Value
A4h	1	WCAP	10
A8h	1	ECNT	Control Field Value
ACh	1	ECNT	3
B0h	1	PCNT	Control Field Value
B4h	1	PCNT	4

The necessary setup of the HTU control packet (see [Section 20.5](#)) for this example is as follows:

```

IHADDR = 0x34 (points to WCAP control field)
IFADDR = 0x70 (points to buffer)
ITCOUNT [frame count = 3] [element count = 3]
IHADDRCT = [DIR: Read HET and write to full address]
           [SIZE: 64 bit]
           [ADDMH: Increment HET address by 16 bytes]
           [ADDMF: post increment full address mode]
           [Any transfer mode]

```

For different applications, which have the transfer direction set for reading the buffer and writing to HET fields, the 64-bit transfer could be used to change the conditional addresses together with a new data field.

## 20.4 Control Registers

**Table 20-10** provides a summary of the registers. The registers support 8-bit, 16-bit, and 32-bit writes. The offset is relative to the associated peripheral select. See the following sections for detailed descriptions of the registers. The base address for the control registers is FFF7 A400h. The address locations not listed, are reserved.

**Table 20-10. Control Register Mapping**

Offset	Acronym	Register Description	Section
00	HTU GC	Global Control Register	<a href="#">Section 20.4.1</a>
04h	HTU CPENA	Control Packet Enable Register	<a href="#">Section 20.4.2</a>
08h	HTU BUSY0	Control Packet Busy Register 0	<a href="#">Section 20.4.3</a>
0Ch	HTU BUSY1	Control Packet Busy Register 1	<a href="#">Section 20.4.4</a>
10h	HTU BUSY2	Control Packet Busy Register 2	<a href="#">Section 20.4.5</a>
14h	HTU BUSY3	Control Packet Busy Register 3	<a href="#">Section 20.4.6</a>
18h	HTU ACPE	Active Control Packet and Error Register	<a href="#">Section 20.4.7</a>
20h	HTU RLBECTRL	Request Lost and Bus Error Control Register	<a href="#">Section 20.4.8</a>
24h	HTU BFINTS	Buffer Full Interrupt Enable Set Register	<a href="#">Section 20.4.9</a>
28h	HTU BFINTC	Buffer Full Interrupt Enable Clear Register	<a href="#">Section 20.4.10</a>
2Ch	HTU INTMAP	Interrupt Mapping Register	<a href="#">Section 20.4.11</a>
34h	HTU INTOFF0	Interrupt Offset Register 0	<a href="#">Section 20.4.12</a>
38h	HTU INTOFF1	Interrupt Offset Register 1	<a href="#">Section 20.4.13</a>
3Ch	HTU BIM	Buffer Initialization Mode Register	<a href="#">Section 20.4.14</a>
40h	HTU RLOSTFL	Request Lost Flag Register	<a href="#">Section 20.4.15</a>
44h	HTU BFINTFL	Buffer Full Interrupt Flag Register	<a href="#">Section 20.4.16</a>
48h	HTU BERINTFL	BER Interrupt Flag Register	<a href="#">Section 20.4.17</a>
4Ch	HTU MP1S	Memory Protection 1 Start Address Register	<a href="#">Section 20.4.18</a>
50h	HTU MP1E	Memory Protection 1 End Address Register	<a href="#">Section 20.4.19</a>
54h	HTU DCTRL	Debug Control Register	<a href="#">Section 20.4.20</a>
58h	HTU WPR	Watch Point Register	<a href="#">Section 20.4.21</a>
5Ch	HTU WMR	Watch Mask Register	<a href="#">Section 20.4.22</a>
60h	HTU ID	Module Identification Register	<a href="#">Section 20.4.23</a>
64h	HTU PCR	Parity Control Register	<a href="#">Section 20.4.24</a>
68h	HTU PAR	Parity Address Register	<a href="#">Section 20.4.25</a>
70h	HTU MPCS	Memory Protection Control and Status Register	<a href="#">Section 20.4.26</a>
74h	HTU MP0S	Memory Protection 0 Start Address Register	<a href="#">Section 20.4.27</a>
78h	HTU MP0E	Memory Protection 0 End Address Register	<a href="#">Section 20.4.28</a>

### 20.4.1 Global Control Register (HTU GC)

**Figure 20-14. Global Control Register (HTU GC) [offset = 00]**

31	25	24	23	17	16
Reserved		VBUSHOLD	Reserved		HTUEN
R-0		R/W-0	R-0		R/W-0
15	9	8	7	1	0
Reserved		DEBM	Reserved		HTURES
R-0		R/W-0	R-0		R/W-0

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

**Table 20-11. Global Control Register (HTU GC) Field Descriptions**

Bit	Field	Value	Description
31-25	Reserved	0	Read returns 0. Writes have no effect.
24	VBUSHOLD	0 1	Hold the VBUS bus The VBUS is not held The VBUSHOLD bit holds the bus used to transfer data between the HTU and the NHET module. When the BUS_BUSY bit is zero then the bus is no longer busy. While the bus is held, requests will still be accepted. They will be acted upon when the VBUSHOLD is zero. Request lost conditions will be detected and interrupts generated if they are enabled.
23-17	Reserved	0	Read returns 0. Writes have no effect.
16	HTUEN	0 1	Transfer Unit Enable Bit The Transfer Unit is disabled The Transfer Unit is enabled  The configuration registers and control packets should be set up first before the HTUEN bit is set to one to it from carrying out unintended bus transactions. If the HTUEN bit is cleared during a frame is transferred, then the frame will be completed before the HTU is disabled.  The HTUEN bit must be cleared and the parity functionality must be enabled (by PARITY_ENA) during the automatic DCP RAM initialization (see Initializing Parity Bits). If HTUEN is one when the initialization is triggered by the system module, then the initialization will not be performed and the HTU operation is not affected. If a one is written to HTUEN during the initialization, then the HTUEN bit will be set but the HTU will not be enabled before the initialization completes.  <b>Note: If HTU is disabled during a frame transfer, then the ongoing current frame will be completed before the HTU module is disabled. If enabled again, then the transfer will restart from the initial frame count for the CP programmed.</b>
15-9	Reserved	0	Read returns 0. Writes have no effect.
8	DEBM	0 1	Debug Mode The Transfer Unit is stopped in debug mode. The HTU will complete the current frame, but not start any new frames. It will also ignore all requests from the HET and not generate any request lost signals. The Transfer Unit continues operation in debug mode <b>Note:</b> Since the HET has also an "ignore suspend" bit, there a several possibilities for the behavior of the HET and HTU in suspend mode.
7-1	Reserved	0	Read returns 0. Writes have no effect.
0	HTURES	0 1	HTU Software Reset The Transfer Unit is not in reset state The Transfer Unit is enabled.  Ongoing element transfers will be completed, before resetting the complete HTU module, similar to a hardware reset. The HTURES bit will also be cleared. The recommended order of operations is: <ul style="list-style-type: none"> <li>Set the software reset bit. This also clears HTUEN.</li> <li>Wait for the HTURES bit to clear.</li> <li>Configure the HTU registers and packets.</li> <li>Set the HTUEN bit to begin operation.</li> </ul>

### 20.4.2 Control Packet Enable Register (HTU CPENA)

This register enables or disables the individual double control packets (DCP).

**Figure 20-15. Control Packet Enable Register (HTU CPENA) [offset = 04h]**

31	Reserved	16
R-0		
15	CPENA	0
R/W-0		

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

**Table 20-12. Control Packet Enable Register (HTU CPENA) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	CPENA		CP Enable Bits Bits (2-x) and (2-x+1) of CPENA control the double control packet (DCP) x (whereby x must be within 0,1,...,7) See <a href="#">Table 20-13</a> for write rules See <a href="#">Table 20-14</a> for read rules

**Table 20-13. CPENA Write Results**

Bit (2-x+1)	Bit (2-x)	Control packets (CP) B and A of DCP x are affected as follows:
0	0	CP B and A are not changed
0	1	CP B is disabled and CP A are enabled simultaneously
1	0	CP B is enabled and CP A are disabled simultaneously
1	1	CP B and CP A are both disabled simultaneously

**Table 20-14. CPENA Read Results**

Bit (2-x+1)	Bit (2-x)	State of DCP:
0	0	The DCP is disabled
0	1	CP B is disabled and CP A is enabled
1	0	CP B is enabled and CP A is disabled
1	1	Cannot be read

- The conditions listed in [Section 20.2.3](#) can automatically disable DCP x. In this case bits (2-x) and (2-x+1) are both automatically set to zero.
- When bits (2-x) and (2-x+1) change from 00 to 01 or from 00 to 10 caused by a write access to CPENA, then old pending requests on the corresponding request line are cleared. This means only new requests which occur **after** this write access cause the first HTU transfer for this DCP. This is **not** the case when **switching** CPs (from 10 to 01 or from 01 to 10).
- CP A and/or CP B of a DCP can be configured to one-shot, circular or auto-switch transfer mode via the TMBA or TMBB bits in the IHADDRCT control packet configuration. If a write access to CPENA occurs during the last frame of a buffer (with frame counter = 1) then the action defined by the write access to CPENA and the action defined by TMBx can contradict. The priority rules for this case are given in [Table 20-1](#).

### 20.4.3 Control Packet (CP) Busy Register 0 (HTU BUSY0)

This register displays the status of individual control packets.

**Figure 20-16. Control Packet (CP) Busy Register 0 (HTU BUSY0) [offset = 08h]**

31	25	24	23	17	16
Reserved		BUSY0A	Reserved		BUSY0B
R-0		R/W-0	R-0		R/W-0
15	9	8	7	1	0
Reserved		BUSY1A	Reserved		BUSY1B
R-0		R/W-0	R-0		R/W-0

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

**Table 20-15. Control Packet (CP) Busy Register 0 (HTU BUSY0) Field Descriptions**

Bit	Field	Value	Description
31-25	Reserved	0	Read returns 0. Writes have no effect.
24	BUSY0A		Busy Flag for CP A of DCP 0
23-17	Reserved	0	Read returns 0. Writes have no effect.
16	BUSY0B		Busy Flag for CP B of DCP 0
15-9	Reserved	0	Read returns 0. Writes have no effect.
8	BUSY1A		Busy Flag for CP A of DCP 1
7-1	Reserved	0	Read returns 0. Writes have no effect.
0	BUSY1B		Busy Flag for CP B of DCP 1

The bit is **set** when the frame on the according control packet **starts** (as shown in the diagram below, there could be a delay between the request and the start of the frame).

The bit is automatically **cleared** at any of the following conditions:

1. At the end of a frame.
2. Writing a 1 to a BUSY bit (of DCP x) if that bit is 1. This will:
  - (a) clear the element counter of DCP x,
  - (b) stop all new element transfers on DCP x
  - (c) clear the busy bit and (4.) disable DCP x in the CPENA register.
  - (d) there is no effect if the BUSY bit is zero.
3. At the conditions listed in [Section 20.2.3, Conditions for Frame Transfer Interruption](#).

A write access to the CPENA register can stop a control packet (CP) in single buffer mode or it can switch to the other CP of a DCP in dual buffer mode. If stopping or switching occurs while a frame runs on the currently active control packet, the CPU can poll the busy bit to determine when it is safe to read the buffer.

### 20.4.4 Control Packet (CP) Busy Register 1 (HTU BUSY1)

This register displays the status of individual control packets.

**Figure 20-17. Control Packet (CP) Busy Register 1 (HTU BUSY1) [offset = 0Ch]**

31	25	24	23	17	16
Reserved		BUSY2A	Reserved		BUSY2B
R-0		R/W-0	R-0		R/W-0
15	9	8	7	1	0
Reserved		BUSY3A	Reserved		BUSY3B
R-0		R/W-0	R-0		R/W-0

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

**Table 20-16. Control Packet (CP) Busy Register 1 (HTU BUSY1) Field Descriptions**

Bit	Field	Value	Description
31-25	Reserved	0	Read returns 0. Writes have no effect.
24	BUSY2A		Busy Flag for CP A of DCP 2
23-17	Reserved	0	Read returns 0. Writes have no effect.
16	BUSY2B		Busy Flag for CP B of DCP 2
15-9	Reserved	0	Read returns 0. Writes have no effect.
8	BUSY3A		Busy Flag for CP A of DCP 3
7-1	Reserved	0	Read returns 0. Writes have no effect.
0	BUSY3B		Busy Flag for CP B of DCP 3

See [Section 20.4.3](#) for more details.



### 20.4.5 Control Packet (CP) Busy Register 2 (HTU BUSY2)

**Figure 20-18. Control Packet (CP) Busy Register 2 (HTU BUSY2) [offset = 10h]**

31	25	24	23	17	16
Reserved		BUSY4A	Reserved		BUSY4B
R-0		R/W-0	R-0		R/W-0
15	9	8	7	1	0
Reserved		BUSY5A	Reserved		BUSY5B
R-0		R/W-0	R-0		R/W-0

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

**Table 20-17. Control Packet (CP) Busy Register 2 (HTU BUSY2) Field Descriptions**

Bit	Field	Value	Description
31-25	Reserved	0	Read returns 0. Writes have no effect.
24	BUSY4A		Busy Flag for CP A of DCP 4
23-17	Reserved	0	Read returns 0. Writes have no effect.
16	BUSY4B		Busy Flag for CP B of DCP 4
15-9	Reserved	0	Read returns 0. Writes have no effect.
8	BUSY5A		Busy Flag for CP A of DCP 5
7-1	Reserved	0	Read returns 0. Writes have no effect.
0	BUSY5B		Busy Flag for CP B of DCP 5

See [Section 20.4.3](#) for more details.

### 20.4.6 Control Packet (CP) Busy Register 3 (HTU BUSY3)

**Figure 20-19. Control Packet (CP) Busy Register 3 (HTU BUSY3) [offset = 14h]**

31	25	24	23	17	16
Reserved	BUSY6A		Reserved		BUSY6B
R-0	R/W-0		R-0		R/W-0
15	9	8	7	1	0
Reserved		BUSY7A	Reserved		BUSY7B
R-0		R/W-0	R-0		R/W-0

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

**Table 20-18. Control Packet (CP) Busy Register 3 (HTU BUSY3) Field Descriptions**

Bit	Field	Value	Description
31-25	Reserved	0	Read returns 0. Writes have no effect.
24	BUSY6A		Busy Flag for CP A of DCP 6
23-17	Reserved	0	Read returns 0. Writes have no effect.
16	BUSY6B		Busy Flag for CP B of DCP 6
15-9	Reserved	0	Read returns 0. Writes have no effect.
8	BUSY7A		Busy Flag for CP A of DCP 7
7-1	Reserved	0	Read returns 0. Writes have no effect.
0	BUSY7B		Busy Flag for CP B of DCP 7

### 20.4.7 Active Control Packet and Error Register (HTU ACPE)

**Figure 20-20. Active Control Packet and Error Register (HTU ACPE) [offset = 18h]**

31	30	29	28	24	23	20	19	16
ERRF	Reserved		ERRETC			Reserved		ERRCPN
R/W-0	R-0		R-0			R-0		R-0
15	14	13	12	8	7	4	3	0
TIPF	BUSBUSY	Rsvd	CETCOUNT			Reserved		NACP
R-0	R-0	R-0	R-0			R-0		R-0

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

**Table 20-19. Active Control Packet and Error Register (HTU ACPE) Field Descriptions**

Bit	Field	Value	Description						
31	ERRF	0 1	Error Flag No error occurred This bit is set when one of the conditions listed at ERRETC is fulfilled and ERRETC and ERRCPN are captured. Once ERRF is set, it is cleared when reading the upper 16-bit word of the ACPE register or the complete 32-bit register. It is also cleared when writing a one to ERRF. ERRF can be used to indicate if ERRETC and ERRCPN contain new unread data.						
30-29	Reserved	0	Read returns 0. Writes have no effect.						
28-24	ERRETC		Error Element Transfer Count If one of the following conditions happens the current element transfer counter of the control packet (specified by ERRCPN) is captured to ERRETC. Please see also Conditions for Frame Transfer Interruption. <ul style="list-style-type: none"> <li>Request Lost Error of control packet specified by ERRCPN. This is independent of the CORL bit.</li> <li>Parity Error of control packet specified by ERRCPN. This requires the parity check to be enabled, but is independent of the COPE bit.</li> <li>Bus Error of control packet specified by ERRCPN.</li> <li>Memory Protection Error of control packet specified by ERRCPN. This requires the memory protection to be enabled.</li> <li>Writing a one to a BUSY bit, which belongs to the control packet specified by ERRCPN, if that bit is one. There is no effect if the BUSY bit is zero.</li> </ul> ERRETC is frozen from being updated until the upper 16-bit word of the ACPE register or the complete 32-bit register is read by the CPU. After this read the HTU will update ERRETC if one of the above conditions is fulfilled again. During debugging, ERRETC stays frozen even when reading the upper 16-bit word or the 32-bit register.						
23-20	Reserved	0	Read returns 0. Writes have no effect.						
19-16	ERRCPIN		Error Control Packet Number If one of the conditions listed at ERRETC happens the number of the control packet, which caused the condition, is captured to ERRCPN. <table border="0"> <tr> <td><b>Control Packet</b></td><td><b>ERRCPN Value</b></td></tr> <tr> <td>CP A of DCP x</td><td>2 x</td></tr> <tr> <td>CP B of DCP x</td><td>2 x+1</td></tr> </table> With x = 0,1,...or 7 ERRCPN is frozen from being updated until the upper 16-bit word of the ACPE register or the complete 32-bit register is read by the CPU. After this read the HTU will update ERRCPN if one of the above conditions is fulfilled again. During debugging, ERRCPN stays frozen even when reading the upper 16-bit word or the 32-bit register. If one of the conditions is fulfilled ERRETC and ERRCPN are updated simultaneously.	<b>Control Packet</b>	<b>ERRCPN Value</b>	CP A of DCP x	2 x	CP B of DCP x	2 x+1
<b>Control Packet</b>	<b>ERRCPN Value</b>								
CP A of DCP x	2 x								
CP B of DCP x	2 x+1								
15	TIPF	0 1	Transfer in Progress Flag No transfers are in progress A transfer is currently active. This bit is the result of a logical OR function of all BUSYxx flags of the 4 BUSYx registers.						

**Table 20-19. Active Control Packet and Error Register (HTU ACPE) Field Descriptions (continued)**

Bit	Field	Value	Description						
14	BUSBUSY	0 1	<p>Bus is Busy</p> <p>0 Bus between NHET and HTU is not busy</p> <p>1 When BUSBUSY is one the bus is busy with a transfer. It is different from TIPF above because BUSBUSY will go low after VBUSHOLD is set to one and no transfers are pending between the HTU and the main memory. TIPF will remain one if a transfer is still pending and VBUSPHOLD is one.</p>						
13	Reserved	0	Read returns 0. Writes have no effect.						
12-8	CETCOUNT		<p>Current Element Transfer Count</p> <p>CETCOUNT shows the current element transfer counter for the frame which is currently processed. If the HTU doesn't currently transfer any frame CETCOUNT is zero.</p> <p>CETCOUNT is updated after the write part of a transfer. There is a period of up to 7 cycles between the time the CETCOUNT is zero and the HTU is finished updating the current DCP (and the CPENA registers if the required conditions are fulfilled).</p>						
7-4	Reserved	0	Read returns 0. Writes have no effect.						
3-0	NACP		<p>Number of Active Control Packet</p> <p>Indicates which CP currently processes a frame.</p> <table border="0"> <thead> <tr> <th>Active or Recent DCP</th> <th>NACP Value</th> </tr> </thead> <tbody> <tr> <td>CP A of DCP x</td> <td>2 x</td> </tr> <tr> <td>CP B of DCP x</td> <td>2 x+1</td> </tr> </tbody> </table> <p>With x = 0,1,...or 7</p> <p>NACP is updated at the time the frame starts on the according CP, and it is updated with a new value when a frame starts on a different CP. Note, that there can be a delay between the request and the start of the frame.</p>	Active or Recent DCP	NACP Value	CP A of DCP x	2 x	CP B of DCP x	2 x+1
Active or Recent DCP	NACP Value								
CP A of DCP x	2 x								
CP B of DCP x	2 x+1								

### 20.4.8 Request Lost and Bus Error Control Register (HTU RLBECTRL)

**Figure 20-21. Request Lost and Bus Error Control Register (HTU RLBECTRL) [offset = 20h]**

31	Reserved										17	16				
											BERINTENA					
											R-0					
											R/W-0					
15	Reserved							9	8	7			1	0		
							CORL		Reserved				RLINTENA			
							R-0		R/W-0				R-0		R/W-0	

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

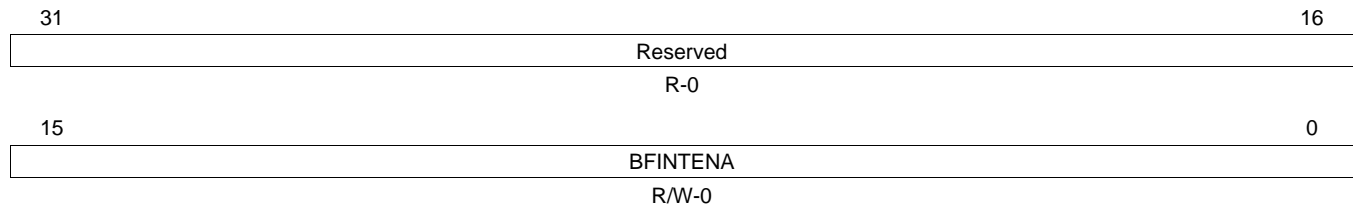
**Table 20-20. Request Lost and Bus Error Control Register (HTU RLBECTRL) Field Descriptions**

Bit	Field	Value	Description
31-17	Reserved	0	Read returns 0. Writes have no effect.
16	BERINTENA	0	Bus Error Interrupt Enable Bit The bus error interrupt is disabled for all DCPs
		1	The bus error interrupt is enabled for all DCPs
15-9	Reserved	0	Read returns 0. Writes have no effect.
8	CORL	0	Continue On Request Lost Error Stop current frame on request lost detection. Please see Conditions for Frame Transfer Interruption
		1	If CORL is one and DCP x is enabled, then DCP x will stay enabled after a request lost condition on DCP x and element transfers will continue.
7-1	Reserved	0	Read returns 0. Writes have no effect.
0	RLINTENA	0	Request Lost Interrupt Enable Bit The request lost interrupt is disabled for all DCPs. Disabling RLINTENA will not clear the flags in the RLOSTFL register.
		1	The request lost interrupt is enabled for all DCPs. If bits are set in the RLOSTFL flag register at the time RLINTENA is (re-) enabled, then the according interrupt(s) will occur (in the order of the priority of the request lines).

### 20.4.9 Buffer Full Interrupt Enable Set Register (HTU BFINTS)

This registers allows to enable the buffer full interrupts for the different control packets. Reading registers BFINTS and BFINTC will return the same bits indicating the status which interrupt is enabled (1) or disabled (0).

**Figure 20-22. Buffer Full Interrupt Enable Set Register (HTU BFINTS) [offset = 24h]**



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

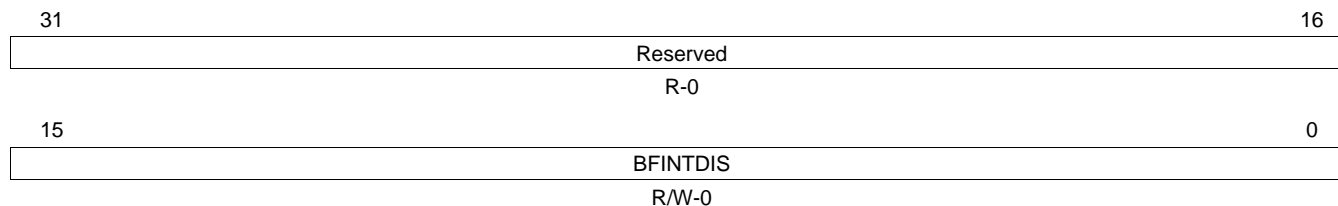
**Table 20-21. Buffer Full Interrupt Enable Set Register (HTU BFINTS) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	BFINTENA	0	Bus Full Interrupt Enable Bits. If the interrupt for CP A of a DCP is enabled, then the interrupt is generated once buffer A is full, i.e. once the frame counter CFTCTA decrements to zero. The same applies for CP B (and CFTCTB). Interrupt disabled. Writing a zero has no effect
		1	Writing to bit (2-x) enables the interrupt for CP A of DCP x. Writing to bit (2-x+1) enables the interrupt for CP B of DCP x.

### 20.4.10 Buffer Full Interrupt Enable Clear Register (HTU BFINTC)

This registers allows to disable the buffer full interrupts for the different control packets. Reading registers BFINTS and BFINTC will return the same bits indicating the status which interrupt is enabled (1) or disabled (0)

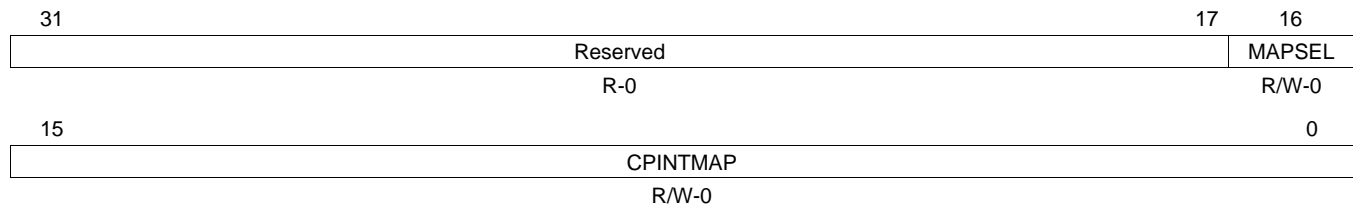
**Figure 20-23. Buffer Full Interrupt Enable Clear Register (HTU BFINTC) [offset = 28h]**



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

**Table 20-22. Buffer Full Interrupt Enable Clear Register (HTU BFINTC) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	BFINTDIS	0	Buffer Full Interrupt Disable Bits Interrupt disabled. Writing a zero no effect
		1	Writing to bit (2-x) disables the interrupt for CP A of DCP x. Writing to bit (2-x+1) disables the interrupt for CP B of DCP x.

**20.4.11 Interrupt Mapping Register (HTU INTMAP)**
**Figure 20-24. Interrupt Mapping Register (HTU INTMAP) [offset = 2Ch]**


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

**Table 20-23. Interrupt Mapping Register (HTU INTMAP) Field Descriptions**

Bit	Field	Value	Description
31-17	Reserved	0	Read returns 0. Writes have no effect.
16	MAPSEL	0	Interrupt Mapping Select Bit If MAPSEL is zero then one bit of CPINTMAP selects one of two interrupt priorities 0 or 1 for the <b>buffer full</b> interrupt for the according CP. The <b>request lost</b> and <b>bus error</b> interrupts of all CPs are set to priority 0, independent of CPINTMAP.
		1	If MAPSEL is one then one bit of CPINTMAP determines if the <b>buffer full</b> , <b>request lost</b> and <b>bus error</b> interrupts of the according CP are assigned either to interrupt line 0 or to 1.
15-0	CPINTMAP	0	CP Interrupt Mapping Bits Interrupt of CP A (bit 2-x) of DCP x is mapped to interrupt line 0. Interrupt of CP B (bit 2-x+1) of DCP x is mapped to interrupt line 0.
		1	Interrupt of CP A (bit 2-x) of DCP x is mapped to interrupt line 1. Interrupt of CP B (bit 2-x+1) of DCP x is mapped to interrupt line 1.

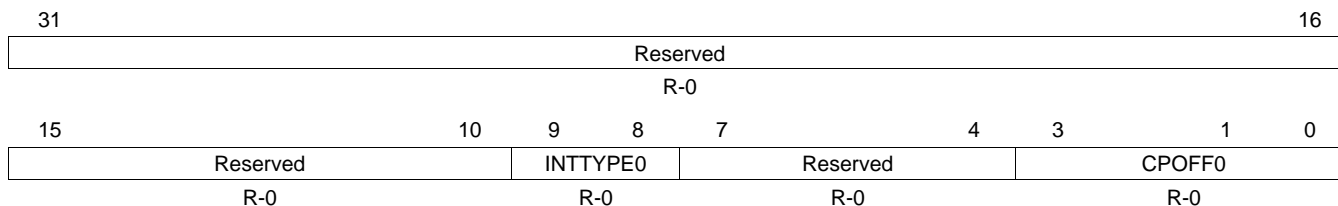


### 20.4.12 Interrupt Offset Register 0 (HTU INTOFF0)

The INTOFF0 register reflects the highest priority interrupt flag bit set in the BERINTFL, RLOSTFL or BFINTFL flag registers with the appropriate CPINTMAP bit set to 0.

- The priority order (from high to low) is: BER, RLOST, buffer-full.
- Interrupts for request lines with lower number (i.e. higher row in the table next page) have higher priority.

**Figure 20-25. Interrupt Offset Register 0 (HTU INTOFF0) [offset = 34h]**



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

**Table 20-24. Interrupt Offset Register 0 (HTU INTOFF0) Field Descriptions**

Bit	Field	Value	Description
31-11	Reserved	0	Read returns 0. Writes have no effect.
10-8	INTTYPE0	0	No interrupt
		1h	Interrupt caused by full buffer on CP/DCP specified by CPOFF0
		2h	RLOST interrupt generated by CP/DCP specified by CPOFF0
		3h	BER interrupt generated by CP/DCP specified by bits CPOFF0
7-5	Reserved	0	Read returns 0. Writes have no effect.
4-0	CPOFF0		CP Offset. Indicates for which control packet the interrupt is pending, which is classified by INTTYPE0 and is assigned to interrupt line 0.
		0	DCP 0, CP A
		1h	DCP 0, CP B
		2h	DCP 1, CP A
		3h	DCP 1, CP B
		4h	DCP 2, CP A
		5h	DCP 2, CP B
		6h	DCP 3, CP A
		7h	DCP 3, CP B
		8h	DCP 4, CP A
		9h	DCP 4, CP B
		Ah	DCP 5, CP A
		Bh	DCP 5, CP B
		Ch	DCP 6, CP A
		Dh	DCP 6, CP B
		Eh	DCP 7, CP A
		Fh	DCP 7, CP B

**NOTE:** Reading CPOFF0 will clear the bit generating the current interrupt from appropriate flag register (BERINTFL, RLOSTFL, or BFINTFL) except when in debug mode where reading CPOFF0 will have no effect on the flag registers.

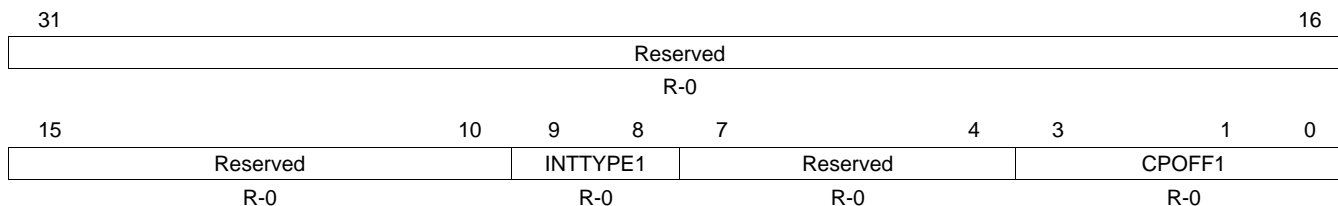
**NOTE:** In order to read INTTYPE0 and CPOFF0 simultaneously always read this register using word or half-word but not using byte accesses.

---

### 20.4.13 Interrupt Offset Register 1 (HTU INTOFF1)

This register is organized identically to the INTOFF0 register. The difference is that INTOFF1 reflects the highest priority interrupt flag bit set in the BERINTFL, RLOSTFL or BFINTFL flag registers with the appropriate CPINTMAP bit set to 1.

**Figure 20-26. Interrupt Offset Register 1 (HTU INTOFF1) [offset = 38h]**



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

**Table 20-25. Interrupt Offset Register 1 (HTU INTOFF1) Field Descriptions**

Bit	Field	Value	Description
31-11	Reserved	0	Read returns 0. Writes have no effect.
10-8	INTTYPE1	0	No interrupt
		1h	Interrupt caused by full buffer on CP/DCP specified by CPOFF1
		2h	RLOST interrupt generated by CP/DCP specified by CPOFF1
		3h	BER interrupt generated by CP/DCP specified by bits CPOFF1
7-5	Reserved	0	Read returns 0. Writes have no effect.
4-0	CPOFF1	0	CP Offset. Indicates for which DCP / CP the interrupt is pending, which is classified by INTTYPE1 and is assigned to interrupt line 1
		1h	DCP 0, CP B
		2h	DCP 1, CP A
		3h	DCP 1, CP B
		4h	DCP 2, CP A
		5h	DCP 2, CP B
		6h	DCP 3, CP A
		7h	DCP 3, CP B
		8h	DCP 4, CP A
		9h	DCP 4, CP B
		Ah	DCP 5, CP A
		Bh	DCP 5, CP B
		Ch	DCP 6, CP A
		Dh	DCP 6, CP B
		Eh	DCP 7, CP A
		Fh	DCP 7, CP B

---

**NOTE:** Reading CPOFF1 will clear the bit generating the current interrupt from appropriate flag register BERINTFL, RLOSTFL, or BFINTFL except when in debug mode where reading CPOFF1 will have no effect on the flag registers.

---



---

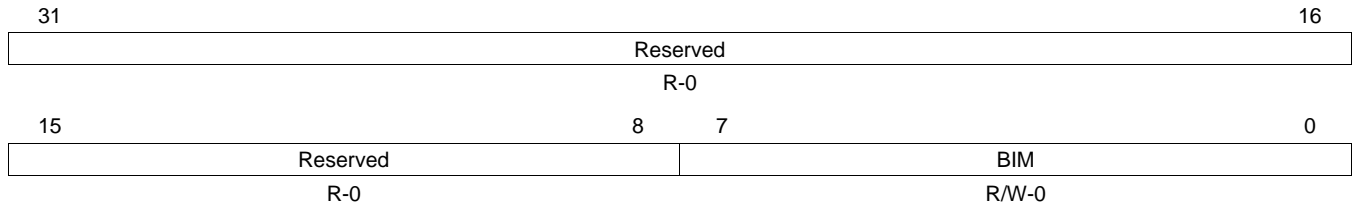
**NOTE:** In order to read INTTYPE1 and CPOFF1 simultaneously always read this register using word or half-word but not using byte accesses.

---

**20.4.14 Buffer Initialization Mode Register (HTU BIM)**

This register enables special applications, where one CP is temporarily disabled, but after having re-enabled the CP, filling the buffer should not start back at its beginning, but should continue after the last element of the previous run.

**Figure 20-27. Buffer Initialization Mode Register (HTU BIM) [offset = 3Ch]**



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

**Table 20-26. Buffer Initialization Mode Register (HTU BIM) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Read returns 0. Writes have no effect.
7-0	BIM		<p>Buffer Initialization Mode</p> <p>The BIM bits and the TMBx bits determine when a buffer is initialized, that means when its initial full address IFADDRx and its initial frame counter IFTCOUNT is used.</p> <p>When initializing (i.e. restarting) a buffer the information in the corresponding initial DCP RAM is loaded to a internal state machine but not to the current DCP RAM (CFADDRx, CFTCTx). The current DCP RAM is updated the first time when the first frame has finished.</p> <p>A buffer is initialized:</p> <ul style="list-style-type: none"> <li>• in circular buffer transfer mode (defined by TMBx) when the end of the buffer is reached.</li> <li>• when CPs are switched or enabled according to Buffer Initialization. The CPENA bits (2-x+1) and (2-x) are changed by write access to CPENA. For the first two rows of the table the change of the CPENA bits could also be the result of the auto switch feature (as defined by TMBx).</li> </ul> <p>BIM bit x only affects DCP x (with x = 0,1,...or 7).</p>

Table 20-27 shows more details on the BIM usage.

**Table 20-27. Buffer Initialization**

Case	Change of CPENA bits (2-x+1) and (2-x)		Action on buffer A or B (of DCP x)		
	Old state <sup>(1)</sup>	New state <sup>(2)</sup>		BIM bit x = 0 (normal mode)	BIM bit x = 1 (special mode)
A	01	10	Switch from CP A to B	Next frame starts at the initial address of buffer B <sup>(3)</sup>	Same as for BIM bit x = 0
B	10	01	Switch from CP B to A	Next frame starts at the initial address of buffer A <sup>(3)</sup>	Same as for BIM bit x = 0
C	01	01	Stay at CP A	Write to CPENA bits (2-x+1) and (2-x) is ignored	Same as for BIM bit x = 0
E	10	10	Stay at CP B	Write to CPENA bits (2-x+1) and (2-x) is ignored	Same as for BIM bit x = 0
E	00	01	Enable CP A	Next frame starts at the initial address of buffer A	Next frame continues at the current address of buffer A
F	00	10	Enable CP B	Next frame starts at the initial address of buffer B	Next frame continues at the current address of buffer B
G	xx	11	Disable both CPs	Stop DCP x	Same as for BIM bit x = 0

<sup>(1)</sup> See read table of CPENA register.

<sup>(2)</sup> See write table of CPENA register.

<sup>(3)</sup> This is regardless of whether the switch is done by a write access to CPENA or by the auto switch feature.

**NOTE:** For cases E and F above, after the last frame of a buffer the HTU sets CFTCTx to zero and CFADDRx to the next address after the buffer. If the DCP was disabled during this state then both CFTCTx and CFADDRx would contain invalid initialization values. Therefore if a DCP should continue at its current address then the software should use one of the following two procedures before it (re-) enables the DCP (as per [Table 20-27](#)):

```

1. If CFTCTx
   ≠ 0
   then set BIM=1
   If CFTCTx = 0 then set BIM=0
2. If CFTCTx ≠ 0 then set
   BIM=1
   If CFTCTx = 0 then {set BIM=1;
set CFTCTx = IFTCOUNT;
set CFADDRx = IFADDRx}

```

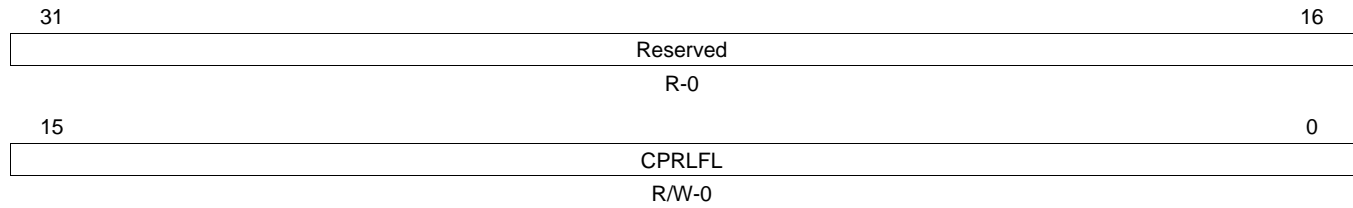
But note that these procedures are only required for the cases E and F and not for all the other cases shown in the table. Also when a buffer reaches its end in circular mode it uses the initial DCP information to restart independently of the BIM setting (assuming it is not temporarily disabled during CFTCTx = 0).

**NOTE:** Similarly care needs to be taken when BIM is set to one and a DCP is enabled for the very first time. Also in this case CFTCTx and CFADDRx usually contain invalid initialization values. The software can either solve this by setting BIM=0 for the first time or setting CFADDRx to IFADDRx and CFTCTx to IFTCOUNT before the DCP is enabled.

**NOTE:** If

- the HTUEN bit is changed to one after the HTU was disabled HTUEN=0
- the CPENA bit pair is 01 or 10 (during this HTUEN change)

then the corresponding BIM bit will decide if the corresponding buffer continues at its initial or current address. Cases E and F in [Table 20-27](#) also apply for this situation. The software should use the procedures explained in the first note before setting HTUEN.

**20.4.15 Request Lost Flag Register (HTU RLOSTFL)**
**Figure 20-28. Request Lost Flag Register (HTU RLOSTFL) [offset = 40h]**


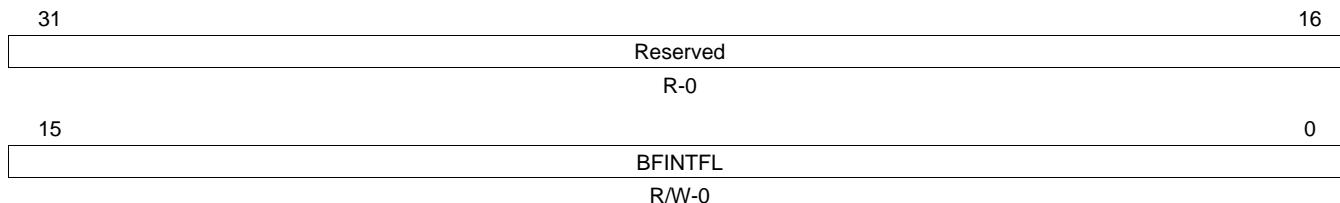
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 20-28. Request Lost Flag Register (HTU RLOSTFL) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	CPRLFL	0	CP Request Lost Flags No request was lost. Writing a zero no effect
		1	If bit (2-x) is set, a request was lost on CP A of DCP x. If bit (2-x+1) is set, a request was lost on CP B of DCP x. Reading from INTOFFx in case of a RLOST interrupt clears the corresponding flag. The state of the flag bit can be polled even if RLINTENA is cleared. <ul style="list-style-type: none"> <li>• Reading CPRLFL will not clear the flags or</li> <li>• Reading from INTOFFx clears the corresponding flag.</li> <li>• Writing a one clears the corresponding flag.</li> </ul>

### 20.4.16 Buffer Full Interrupt Flag Register (HTU BFINTFL)

**Figure 20-29. Buffer Full Interrupt Flag Register (HTU BFINTFL) [offset = 44h]**



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

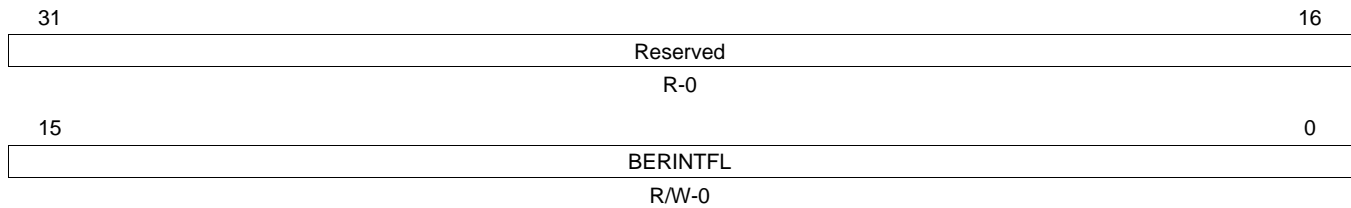
**Table 20-29. Buffer Full Interrupt Flag Register (HTU BFINTFL) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	BFINTFL	0 1	<p>Buffer Full Interrupt Flags</p> <p>0 No buffer full condition detected. Writing a zero no effect</p> <p>1 If bit (2-x) is set, a buffer full condition on CP A of DCP x has been detected. If bit (2-x+1) is set, a buffer full condition on CP B of DCP x has been detected.</p> <p>The BFINTFL flag is set after the last frame finishes on the corresponding buffer regardless of whether the buffer is configured to one shot, circular or auto-switch mode. If BFINTFL is set in circular mode, then a circular overrun has occurred on the corresponding buffer. This can be used to indicate whether the buffer section after the frozen full address contains valid data or not.</p> <p>Reading from INTOFFx in case of a buffer-full interrupt clears the corresponding flag. The state of the flag bit can be polled even if the corresponding interrupt enable bit is cleared.</p> <ul style="list-style-type: none"> <li>• Reading BFINTFL will not clear the flags or</li> <li>• Reading INTOFFx will clear the corresponding flags or</li> <li>• Writing a one clears the corresponding flag.</li> </ul>

**20.4.17 BER Interrupt Flag Register (HTU BERINTFL)**

A bus error interrupt results due to an address error or a timeout condition on the main memory access. A bus error will stop the frame transfer. Please see [Section 20.2.3, Conditions for Frame Transfer Interruption](#)

**Figure 20-30. BER Interrupt Flag Register (HTU BERINTFL) [offset = 48h]**



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

**Table 20-30. BER Interrupt Flag Register (HTU BERINTFL) Field Descriptions**

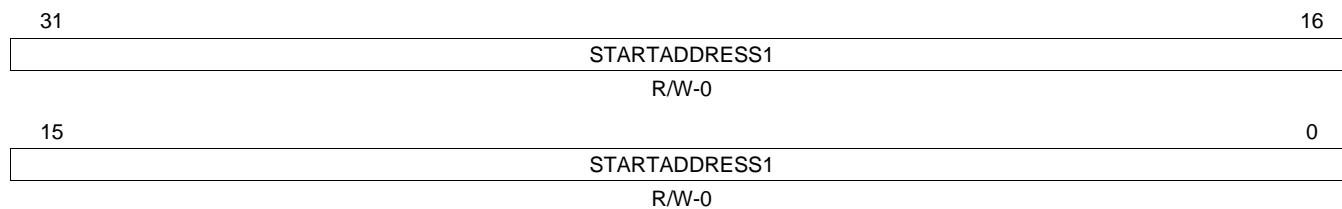
Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	BERINTFL	0	Bus Error Interrupt Flags No bus error condition detected. Writing a zero no effect
		1	If bit (2-x) is set, then a BER interrupt is pending on CP A of DCP x. If bit (2-x+1) is set, then a BER interrupt is pending on CP B of DCP x. The state of the flag bit can be polled even if BERINTENA is cleared. <ul style="list-style-type: none"> <li>• Reading BERINTFL will not clear the flags or</li> <li>• Reading from INTOFFx in case of a BER interrupt clears the corresponding flag or</li> <li>• Writing a one clears the corresponding flag.</li> </ul>



### 20.4.18 Memory Protection 1 Start Address Register (HTU MP1S)

This register configures the start address of memory protection region 1.

**Figure 20-31. Memory Protection 1 Start Address Register (HTU MP1S) [offset = 4Ch]**



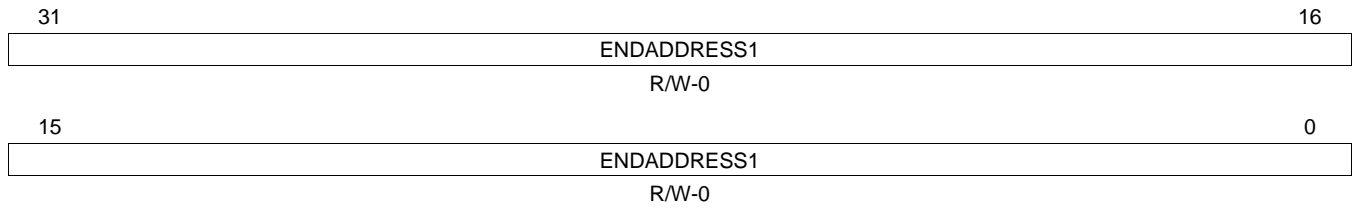
LEGEND: R = Read only; -n = value after reset

**Table 20-31. Memory Protection 1 Start Address Register (HTU MP1S) Field Descriptions**

Bit	Field	Value	Description
31-0	STARTADDRESS1		The start address defines at which main memory address the region begins. A memory protection error will be triggered, if the HTU accesses an address smaller than STARTADDRESS1 and the MPCS bit REG01ENA register is configured accordingly.

**20.4.19 Memory Protection 1 End Address Register (HTU MP1E)**

**Figure 20-32. Memory Protection 1 End Address Register (HTU MP1E) [offset = 50h]**



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

**Table 20-32. Memory Protection 1 End Address Register (HTU MP1E) Field Descriptions**

Bit	Field	Value	Description
31-0	ENDADDRESS1		The end address defines at which address the region ends. A memory protection error will be triggered, if the HTU accesses an address bigger than ENDADDRESS1 and the register bit REG01ENA is configured accordingly.

### 20.4.20 Debug Control Register (HTU DCTRL)

This register allows to create watchpoints on access to a certain location. It is intended to help debug the application execution during program development.

**Figure 20-33. Debug Control Register (HTU DCTRL) [offset = 54h]**

31	28	27	24	23	17	16	
Reserved		CPNUM		Reserved		HTUDBGS	
R-0		R-0		R-0		R/W-0	
15						1	0
Reserved						DBREN	
R-0						R/W-0	

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

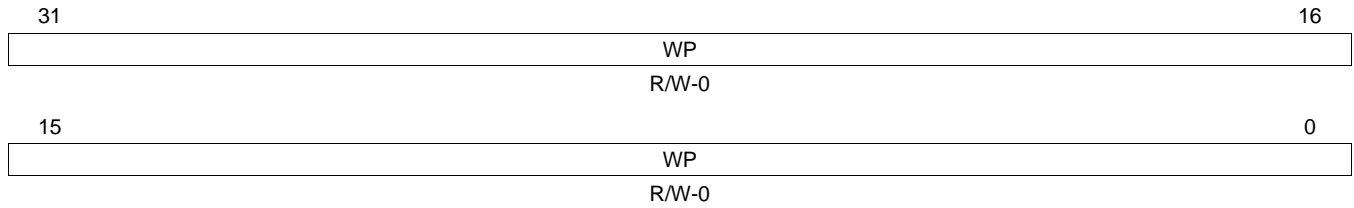
**Table 20-33. Debug Control Register (HTU DCTRL) Field Descriptions**

Bit	Field	Value	Description
31-28	Reserved	0	Read returns 0. Writes have no effect.
27-24	CPNUM	0 1h 2h Eh Fh	CP Number. These bit fields indicate the CP which should cause the watch point to match. CP A of DCP0 CP B of DCP0 CP A of DCP1 CP A of DCP7 CP B of DCP7
23-17	Reserved	0	Read returns 0. Writes have no effect.
16	HTUDBGS	0 1	HTU Debug Status. When the main memory address is equal to the unique address defined by WPR, or lies in the specified range resulting from WMR, then the HTUDBGS is set. If in addition DBREN is set, then the application code execution will be stopped. A one must be written to this bit in order to clear it and to release the CPU from debug halting state. Read: No watch point condition was detected. Write: No effect. Read: A watch point condition was detected. Write: Clears the bit.
15-1	Reserved	0	Read returns 0. Writes have no effect.
0	DBREN		Debug Request Enable If a watch point matches and DBREN is set, then the application code execution will be stopped. This bit can only be set or cleared when in debug mode. This bit and all other bits of the DCTRL, WPR and WMR registers are reset by the test reset (nTRST) but not by the normal device reset.

**20.4.21 Watch Point Register (HTU WPR)**

This register defines the main memory address of the watchpoint.

**Figure 20-34. Watch Point Register (HTU WPR) [offset = 58h]**



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

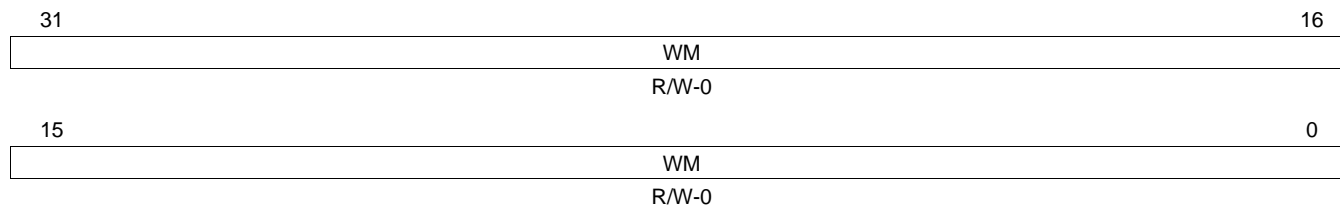
**Table 20-34. Watch Point Register (HTU WPR) Field Descriptions**

Bit	Field	Value	Description
31-0	WP		<p>Watch Point Register</p> <p>A 32-bit address can be programmed into this register as a watch point. The WPR register is used along with the Watch Mask Register ( WMR). When the main memory address is equal to the unique address defined by WPR, or lies in the specified range resulting from WMR, then the HTUDBGS is set. If in addition DBREN is set, then the application code execution is stopped.</p> <p>This register can only be programmed during debug mode. This register and all other bits of the DCTRL and WMR registers are reset by the test reset (nTRST) but not by the normal device reset.</p>

### 20.4.22 Watch Mask Register (HTU WMR)

This register defines a mask of the main memory address of the watchpoint. It can be used to define a memory range in conjunction with the WPR register.

**Figure 20-35. Watch Mask Register (HTU WMR) [offset = 5Ch]**



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

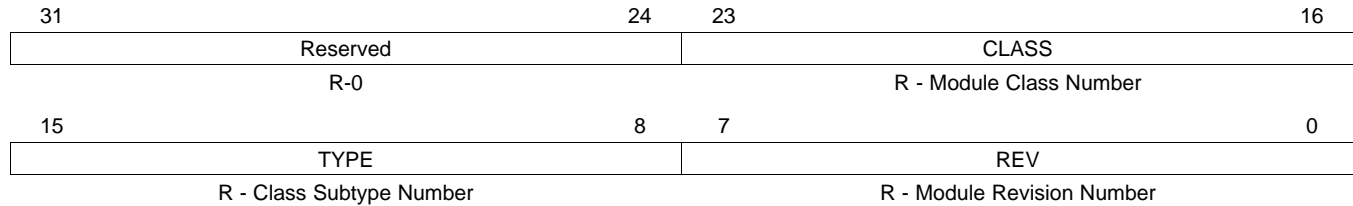
**Table 20-35. Watch Mask Register (HTU WMR) Field Descriptions**

Bit	Field	Value	Description
31-0	WM		<p>Watch Mask Register</p> <p>Setting a bit in the WMR register to one has the effect of masking the corresponding bit in of the main memory address, so that this bit is ignored for the address comparison.</p> <p>This register can only be programmed during debug mode. This register and all other bits of the DCTRL and WPR registers are reset by the test reset (nTRST) but not by the normal device reset.</p>

### 20.4.23 Module Identification Register (HTU ID)

This register is for TI internal purposes and allows to keep track of the HTU module version on different devices.

**Figure 20-36. Module Identification Register (HTU ID) [offset = 60h]**



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

**Table 20-36. Module Identification Register (HTU ID) Field Descriptions**

Bit	Field	Value	Description
31-24	Reserved	0	Read returns 0. Writes have no effect.
23-16	CLASS		Module Class This field defines the module class number as read-only constant value for the HTU module. Writes have no effect.
15-8	TYPE		Subtype within a Class This field defines the subtype within a class as read-only constant value for the HTU module. Writes have no effect.
7-0	REV		Module Revision Number This field defines the module revision number as read-only constant value for the HTU module. Writes have no effect.

### 20.4.24 Parity Control Register (HTU PCR)

**Figure 20-37. Parity Control Register (HTU PCR) [offset = 64h]**

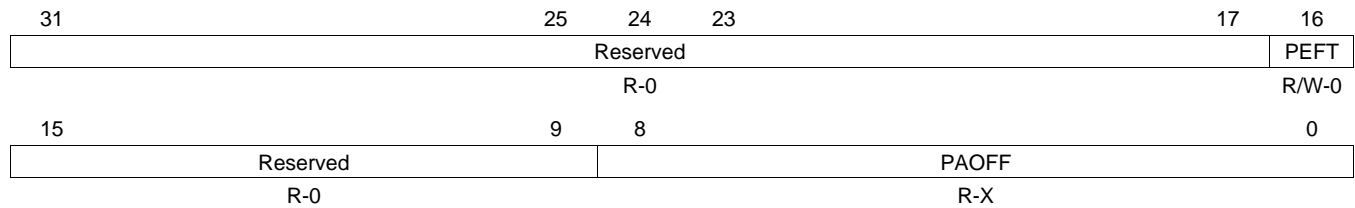
31	25	24	23	17	16
Reserved					COPE
R-0					R/W-0
15	9	8	7	4	3
Reserved		TEST	Reserved		PARITY_ENA
R-0		R/W-0	R-0		R/W-5h

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

**Table 20-37. Parity Control Register (HTU PCR) Field Descriptions**

Bit	Field	Value	Description
31-17	Reserved	0	Read returns 0. Writes have no effect.
16	COPE	0	Continue on Parity Error The HTU performs parity checks every time it reads the RAM section of DCP x (with x = 0, 1,... or 7), before the next frame (of DCP x) is started. If a parity error is detected during this read access and if the parity check is enabled, then the frame will not be started and DCP x will be automatically disabled in the CPENA register. If a master different than the HTU (e.g. CPU) reads the RAM section of DCP x and a parity error is detected during this read access, while the parity check is enabled, then the DCP x will automatically be disabled in the CPENA register. If a frame is active on DCP x during this read access, then in addition the element counter of DCP x is cleared and all new element transfers on DCP x are stopped and the active busy bit of DCP x is cleared.
		1	The difference to COPE=0 is, that the data transfer on a active DCP continues after a parity error was detected on this DCP. So neither the DCP with the parity error will be disabled nor the frame will be stopped.
15-9	Reserved	0	Read returns 0. Writes have no effect.
8	TEST	0	Test. When this bit is set, the parity bits are mapped into the peripheral RAM frame to make them accessible by the CPU. Parity bits are not memory mapped
		1	Parity bits are memory mapped
7-4	Reserved	0	Read returns 0. Writes have no effect.
3-0	PARITY_ENA	5h	Enable/Disable Parity Checking. This bit field enables or disables the parity check on read operations and the parity calculation on write operations. If parity checking is enabled and a parity error is detected, then the PEFT flag is set, PAOFF is captured if it is not currently frozen and an interrupt is generated if it is enabled. Parity check is disabled
		All Others	Parity check is enabled <b>Note:</b> It is recommended to write a "5h" to enable error detection, to guard against single bit changes from flipping PARITY_ENA to a disable state.

### 20.4.25 Parity Address Register (HTU PAR)

**Figure 20-38. Parity Address Register (HTU PAR) [offset = 68h]**


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

**Table 20-38. Parity Address Register (HTU PAR) Field Descriptions**

Bit	Field	Value	Description
31-10	Reserved	0	Read returns 0. Writes have no effect.
16	PEFT	0 1	Parity Error Fault Flag. This bit is set, when the HTU detects a parity error and parity checking is enabled. No fault detected Fault detected <b>Note:</b> Once PEFT is set, a read access to the lower 16 bits or to the complete 32-bit HTUPAR register will clear the PEFT flag in non-debug mode. Another possibility to clear PEFT is to write a 1 to the PEFT bit.
15-9	Reserved	0	Read returns 0. Writes have no effect.
8-0	PAOFF		Parity Error Address Offset. This bit field holds the address of the first parity error, which is detected in the DCP RAM. PAOFF provides the offset address of the erroneous byte counted from the beginning of the DCP memory. This error address is frozen from being updated until a read access to the lower 16 bits or to the complete 32-bit HTUPAR register happens. During debug mode, this address is frozen even when read. <b>Note:</b> The Parity Error Address bits will not be reset, neither by PORRST nor by any other reset source.



### 20.4.26 Memory Protection Control and Status Register (HTU MPCS)

**Figure 20-39. Memory Protection Control and Status Register (HTU MPCS) [offset = 70h]**

31	28	27	24				
Reserved		CPNUM0					
R-0		R-0					
23	18	17	16				
Reserved		MPEFT1	MPEFT0				
R-0		R/W-0	R/W-0				
15	12	11	8				
Reserved		CPNUM1					
R-0		R-0					
7	6	5	4	3	2	1	0
Reserved		INT ENA01	ACCR01	REG01ENA	INT ENA0	ACCR0	REG0ENA
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

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

**Table 20-39. Memory Protection Control and Status Register (HTU MPCS) Field Descriptions**

Bit	Field	Value	Description
31-28	Reserved	0	Read returns 0. Writes have no effect.
27-24	CPNUM0	0 1h 2h Eh Fh	Control Packet Number for single memory protection region configuration. CPNUM0 holds the number of the CP, which has caused the first memory protection error when only one memory protection region is used. This number is not updated for multiple access violations until it is read by the CPU. During debug mode, CPNUM0 is frozen even when read. CP A of DCP0 CP B of DCP0 CP A of DCP1 CP A of DCP7 CP B of DCP7
23-18	Reserved	0	Read returns 0. Writes have no effect.
17	MPEFT1	0 1	Memory Protection Error Fault Flag 1. This bit is set, when the HTU performs an access outside the region defined by the MP0S and MP0E and the MP1S and MP0E registers, when the access violates the rights defined by ACCR01 and when the REG01ENA bit is set. No fault detected. Writing a zero has no effect Fault detected. Writing a one will clear the bit.
16	MPEFT0	0 1	Memory Protection Error Fault Flag 0. This bit is set, when the HTU performs an access outside the region defined by the MP0S and MP0E registers, when the access violates the rights defined by ACCR and when the REG0ENA bit is set. No fault detected. Writing a zero has no effect Fault detected. Writing a one will clear the bit.
15-12	Reserved	0	Read returns 0. Writes have no effect.
11-8	CPNUM1	0 1h 2h Eh Fh	Control Packet Number for single memory protection region configuration. CPNUM1 holds the number of the CP, which has caused the first memory protection error when only one memory protection region is used. This number is not updated for multiple access violations until it is read by the CPU. During debug mode, CPNUM1 is frozen even when read. CP A of DCP0 CP B of DCP0 CP A of DCP1 CP A of DCP7 CP B of DCP7
7-6	Reserved	0	Read returns 0. Writes have no effect.

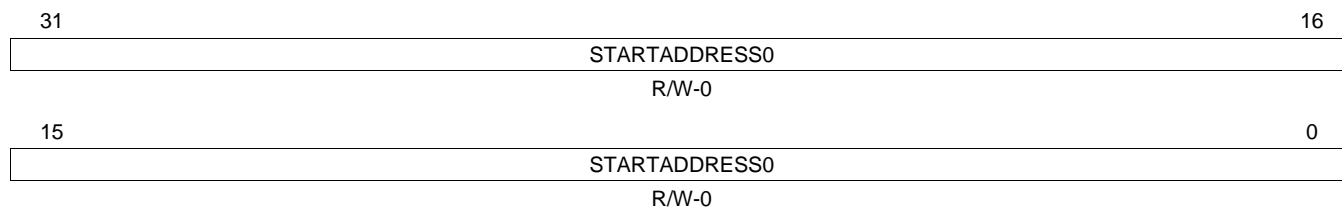
**Table 20-39. Memory Protection Control and Status Register (HTU MPCS) Field Descriptions (continued)**

Bit	Field	Value	Description
5	INTENA01		Interrupt Enable 01. This bit needs to be set when working with two memory mapped regions and a error should be generated to the ESM module on an access violation.
		0	Error signalling is disabled
4	ACCR01	1	Error signalling is enabled
			Access Rights 01. This bit defines the access rights for the HTU for accesses outside the region defined by the MP0S and MP0E and the MP1S and MP0E registers.
0		0	HTU read access is allowed but write access will be signaled
		1	Any access performed by the HTU is forbidden and will be signaled
3	REG01ENA		Region Enable 01. This bit needs to be set when working with two memory mapped regions. <b>REG0ENA must be set to zero</b> if this bit is set to a one. Memory region 0 must be less than memory region 1.
		0	The protection outside the memory region defined by the MP0S and MP0E and the MP1S and MP0E registers is not enabled. This means the HTU can access any implemented memory space. REG0ENA could still enabled to give protection outside the MP0S : MP0E region.
1		1	The protection outside the memory region defined by the MP0S and MP0E and the MP1S and MP0E registers is enabled. This means the HTU can perform any access within the regions, but if it attempts to perform a forbidden access outside of both of the regions (according to the ACCR01 configuration), the access is signaled by the MPEFT1 flag. The number of the CP, which has caused the memory protection error, is captured to CPNUM1 if it is not currently frozen and an error is generated if it is enabled.
			Interrupt Enable 0. This bit needs to be set when working with one memory mapped region and a error should be generated to the ESM module on an access violation.
2	INTENA0	0	Error signalling is disabled
		1	Error signalling is enabled
1	ACCR		Access Rights 0. This bit defines the access rights for the HTU for accesses outside the region defined by the MP0S and MP0E registers for a single memory protection region configuration.
		0	HTU read access is allowed but write access will be signaled
0	REG0ENA	1	Any access performed by the HTU is forbidden and will be signaled
			Region Enable 0
0		0	The protection outside the memory region defined by the MP0S and MP0E registers is not enabled. This means the HTU can access any implemented memory space.
		1	The protection outside the memory region defined by the MP0S and MP0E registers is enabled. This means the HTU can perform any access within the region, but if it attempts to perform a forbidden access outside the region (according to the ACCR configuration), the access is signaled by the MPEFT0 flag, the number of the CP, which has caused the memory protection error, is captured to CPNUM0 if it is not currently frozen and an error is generated if it is enabled.

### 20.4.27 Memory Protection Start Address Register 0 (HTU MP0S)

This register configures the start address of memory protection region 0

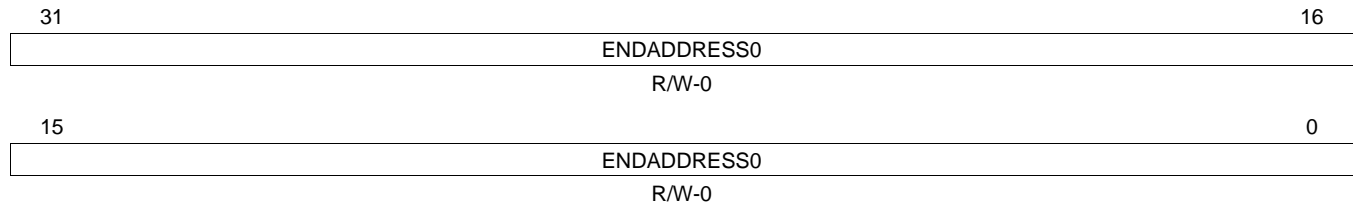
**Figure 20-40. Memory Protection Start Address Register 0 (HTU MP0S) [offset = 74h]**



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

**Table 20-40. Memory Protection 0 Start address Register (HTU MP0S) Field Descriptions**

Bit	Field	Value	Description
31-0	ISTARTADDRESS0		The start address defines at which main memory address the region begins. A memory protection error will be triggered, if the HTU accesses an address smaller than STARTADDRESS0 and the MPCS register is configured accordingly.

**20.4.28 Memory Protection End Address Register (HTU MP0E)**
**Figure 20-41. Memory Protection End Address Register (HTU MP0E) [offset = 78h]**


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

**Table 20-41. Memory Protection End Address Register (HTU MP0E) Field Descriptions**

Bit	Field	Value	Description
31-0	ENDADDRESS0		The end address defines at which address the region ends. A memory protection error will be triggered, if the HTU accesses an address bigger than ENDADDRESS0 and the register bit MPCS register is configured accordingly.

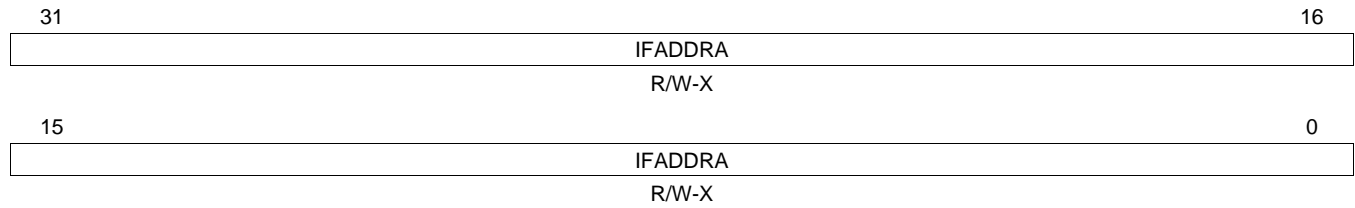
## 20.5 Double Control Packet Configuration Memory

All bits marked "reserved" are implemented in RAM and will be initialized to unknown values after power on. Reserved locations can be written and read but should be written with zero to ensure future compatibility. The HTU RAM can be cleared with the system RAM initialization function.

[Table 20-42](#) provides a summary of the memory configuration. The base address for DCPs is FF4E0000h.

**Table 20-42. Double Control Packet Memory Map**

Offset	Acronym	Register Description	Section
000	HTU IFADDRA	Initial Full Address A Register	<a href="#">Section 20.5.1</a>
004h	HTU IFADDRB	Initial Full Address B Register	<a href="#">Section 20.5.2</a>
008h	HTU IHADDRCT	Initial NHET Address and Control Register	<a href="#">Section 20.5.3</a>
00Ch	HTU ITCOUNT	Initial Transfer Count Register	<a href="#">Section 20.5.4</a>
100h	HTU CFADDRA	Current Full Address A Register	<a href="#">Section 20.5.5</a>
104h	HTU CFADDRB	Current Full Address B Register	<a href="#">Section 20.5.6</a>
108h	HTU CFCOUNT	Current Frame Count Register	<a href="#">Section 20.5.7</a>

**20.5.1 Initial Full Address A Register (HTU IFADDRA)**
**Figure 20-42. Initial Full Address A Register (HTU IFADDRA)**


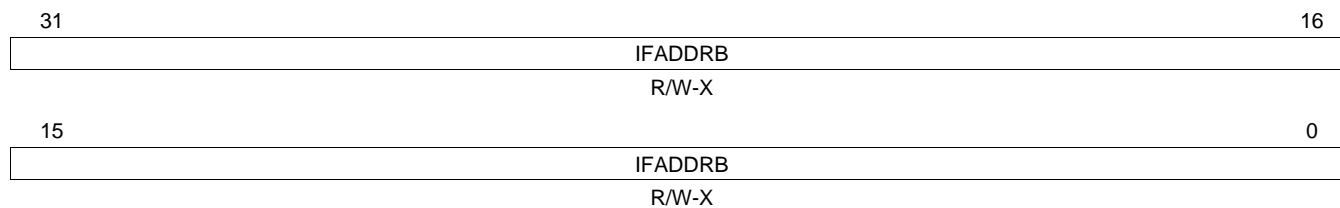
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset; X = Unknown

**Table 20-43. Initial Full Address A Register (HTU IFADDRA) Field Descriptions**

Bit	Field	Value	Description
31-0	IFADDRA		Initial Address of Buffer A in main memory. Initial (byte) address of buffer A placed in the main memory address range. Bits 0 and 1 are ignored by the logic, due to 32 bit alignment.

## 20.5.2 Initial Full Address B Register (HTU IFADDRB)

**Figure 20-43. Initial Full Address B Register (HTU IFADDRB)**



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

**Table 20-44. Initial Full Address B Register (HTU IFADDRB) Field Descriptions**

Bit	Field	Value	Description
31-0	IFADDRB		Initial Address of Buffer B in main memory. Initial (byte) address of buffer B placed in the main memory address range. Bits 0 and 1 are ignored by the logic, due to 32 bit alignment.

### 20.5.3 Initial NHET Address and Control Register (HTU IHADDRCT)

**Figure 20-44. Initial NHET Address and Control Register (HTU IHADDRCT)**

31	24	23	22	21	20	19	18	17	16
Reserved			DIR	SIZE	ADDMH	ADDMF	TMBA	TMBB	
R-0			R/W-X	R/W-X	R/W-X	R/W-X	R/W-X	R/W-X	
15	13	12					2	1	0
Reserved		IHADDR					Reserved		
R-0		R/W-X					R-0		

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

**Table 20-45. Initial NHET Address and Control Register (HTU IHADDRCT) Field Descriptions**

Bit	Field	Value	Description
31-24	Reserved	0	Read returns 0. Writes have no effect.
23	DIR	0	NHET address is read and main memory address is written
		1	Main memory address is read and NHET address is written
22	SIZE	0	32-bit transfer
		1	64-bit transfer 64-bit transfer examples: If the NHET address points to the NHET instruction Control Field (CF), then the CF and Data Field (DF) will be transferred. If the NHET address points to the Program Field (PF), then the PF and CF will be transferred.
21	ADDMH	0	Addressing Mode NHET Address. This bit determines the NHET address index from one to the next element of a frame. Increment by 16 bytes <b>Examples:</b> If the initial NHET address points to data field of instruction (n). Then the NHET fields to be transferred by the elements of a frame are: data field of instruction (n), data field of instruction (n+1), data field of instruction (n+2) and so on. If the initial NHET address points to control field of instruction (n), then the NHET fields to be transferred by the elements of a frame are: control field of instruction (n), control field of instruction (n+1), control field of instruction (n+2) and so on.
		1	Increment by 8 bytes This mode is intended to be used together with the 64-bit transfer size to load short NHET instruction blocks into the NHET RAM. So the sequence of transferred 64-bit elements could be: [PF and CF of instruction (n)], [DF and RF of instruction (n)], [PF and CF of instruction (n+1)], [DF and RF of instruction (n+1)] and so on.
20	ADDFM	0	Addressing Mode Main Memory Address Post-increment <b>Note:</b> When post-increment is selected the HTU will automatically increment by 4 bytes for a 32-bit data size and by 8 bytes for a 64-bit data size.
		1	Constant
19-18	TMBA	0	Transfer Mode for Buffer A One Shot buffer mode
		1h	Circular buffer mode
		All Others	Auto Switch mode
17-16	TMBB	0	Transfer Mode for Buffer B One Shot buffer mode
		1h	Circular buffer mode
		All Others	Auto Switch mode
15-13	Reserved	0	Read returns 0. Writes have no effect.

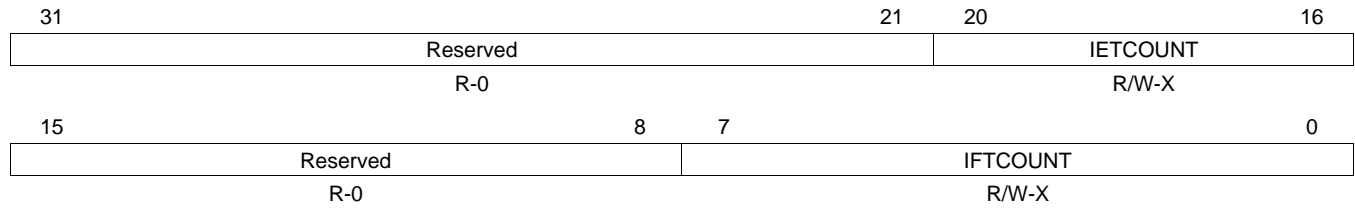


**Table 20-45. Initial NHET Address and Control Register (HTU IHADDRCT) Field Descriptions (continued)**

Bit	Field	Value	Description
12-2	IHADDR		<p>Initial NHET Address</p> <p>The initial NHET Address points to the NHET field, which is the first element of the frame. The NHET address (Bits 12:2) increments by one for each 32-bit NHET field and starts with zero at the first 32-bit field in the NHET RAM.</p> <p><b>Note:</b> When the HTU addresses the NHET RAM it uses only the number of address bits required for the actual NHET RAM size. If the NHET address exceeds the actual NHET RAM size, the unused MSB bits of the address will be ignored and the address rolls over to the start of the NHET RAM.</p>
1-0	Reserved	0	Read returns 0. Writes have no effect.

### 20.5.4 Initial Transfer Count Register (HTU ITCOUNT)

**Figure 20-45. Initial Transfer Count Register (HTU ITCOUNT)**



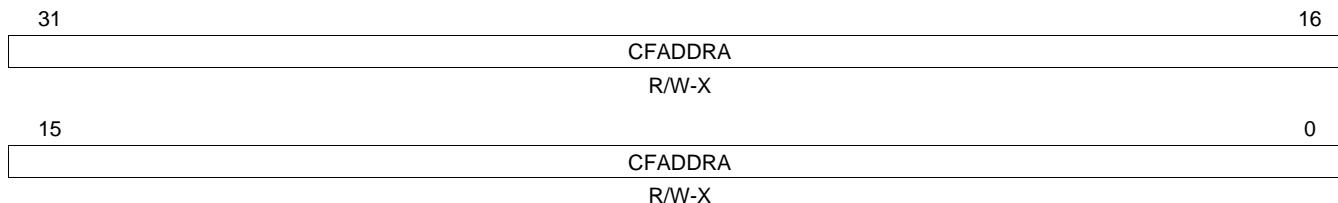
LEGEND: R = Read only; -n = value after reset; X = Unknown

**Table 20-46. Initial Transfer Count Register (HTU ITCOUNT) Field Descriptions**

Bit	Field	Value	Description
31-21	Reserved	0	Read returns 0. Writes have no effect.
20-16	IETCOUNT		Initial Element Transfer Count Defines the number of element transfers.
15-8	Reserved	0	Read returns 0. Writes have no effect.
7-0	IFTCOUNT		Initial Frame Transfer Count Defines the number of frame transfers.

### 20.5.5 Current Full Address A Register (HTU CFADDRA)

**Figure 20-46. Current Full Address A Register (HTU CFADDRA)**



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

**Table 20-47. Current Full Address A Register (HTU CFADDRA) Field Descriptions**

Bit	Field	Value	Description
31-0	CFADDRA		<p>Current (byte) Address of Buffer A</p> <p>The current main memory address register is updated at the end of each frame. Therefore it points to the start address of the frame, which is the next to transfer, if currently no frame is transferred on this DCP. For an ongoing frame transfer, it points to the start address of this frame. After the last element of a buffer was transferred it will point to the buffer end address plus 0x4.</p> <p>The main purpose of the current full address registers for buffer A and buffer B (see next section) is to enable the software to find out the recently transferred element in the frozen buffer while the address of the active buffer increments.</p> <p><b>Note:</b> A frame can be automatically stopped if any of the events listed in Conditions for Frame Transfer Interruption happens. If a frame is stopped before it could complete, then the current full address register is not updated and it will point to the start of the bad frame after the DCP was automatically disabled.</p>

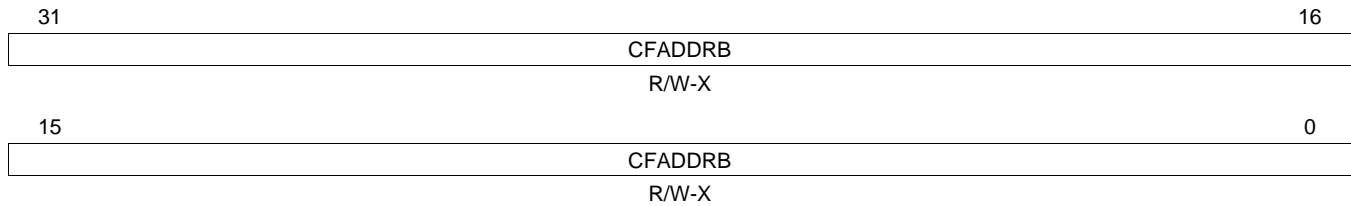
To transfer the first frame of buffer x the information in the corresponding initial DCP RAM (IFADDRx, IHADDRCT, ITCOUNT) is loaded to an internal state machine but not to the current DCP RAM (CFADDRx, CFTCTx).

This is valid for all of the following modes:

- Buffer x has reached its end in circular mode and rolls back to its start address.
- CP x is enabled by a CPENA access (and corresponding BIM bit is zero).
- A CPENA access or auto-switch mode causes a switch from CP y to CP x

This means after starting the transfer to/from buffer x CFADDRx and CFTCTx is not updated before the end of the first frame. So before the software switches from CP y to CP x using a write access to the CPENA register it needs to initialize CFADDRx, CFTCTx. This allows the software to find out if the next request on CP x after the switching to CP x was delayed or never occurring.

## 20.5.6 Current Full Address B Register (HTU CFADDRB)

**Figure 20-47. Current Full Address B Register (HTU CFADDRB)**


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

**Table 20-48. Current Full Address B Register (HTU CFADDRB) Field Descriptions**

Bit	Field	Value	Description
31-0	CFADDRB		<p>Current (byte) Address of Buffer B</p> <p>The current main memory address register is updated at the end of each frame. Therefore it points to the start address of the frame, which is the next to transfer, if currently no frame is transferred on this DCP. If currently a frame is transferred, then it points to the start address of this frame. After the last element of a buffer was transferred it will point to the buffer end address plus 0x4.</p> <p>The main purpose of the current full address registers for buffer A and buffer B (see next section) is to enable the software to find out the recently transferred element in the frozen buffer while the address of the active buffer increments.</p> <p><b>Note:</b> A frame can be automatically stopped if any of the events listed in Conditions for Frame Transfer Interruption happens. If a frame is stopped before it could complete, then the current full address register is not updated and it will point to the start of the bad frame after the DCP was automatically disabled.</p>

To transfer the first frame of buffer x the information in the corresponding initial DCP RAM (IFADDRx, IHADDRCT, ITCOUNT) is loaded to an internal state machine but not to the current DCP RAM (CFADDRx, CFTCTx).

This is valid for all of the following modes:

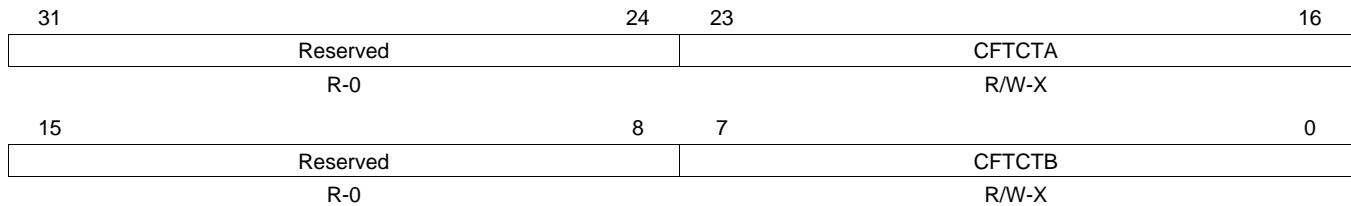
- Buffer x has reached it's end in circular mode and rolls back to its start address.
- CP x is enabled by a CPENA access (and corresponding BIM bit is zero).
- A CPENA access or auto-switch mode causes a switch from CP y to CP x

This means after starting the transfer to/from buffer x CFADDRx and CFTCTx is not updated before the end of the first frame. So before the software switches from CP y to CP x using a write access to the CPENA register it needs to initialize CFADDRx, CFTCTx. This allows the software to find out if the next request on CP x after the switching to CP x was delayed or never occurring.

### 20.5.7 Current Frame Count Register (HTU CFCOUNT)

The current frame count register enables the software to find out the recent element in the frozen buffer while the counter of the active buffer decrements.

**Figure 20-48. Current Frame Count Register (HTU CFCOUNT)**



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

**Table 20-49. Initial Transfer Count Register (HTU CFCOUNT) Field Descriptions**

Bit	Field	Value	Description
31-24	Reserved	0	Read returns 0. Writes have no effect.
23-16	CFTCTA		Current Frame Transfer Count for CP A. It is updated at the end of each frame.
15-8	Reserved	0	Read returns 0. Writes have no effect.
7-0	CFTCTB		Current Frame Transfer Count for CP B. It is updated at the end of each frame.

## 20.6 Examples

### 20.6.1 Application Examples for Setting the Transfer Modes of CP A and B of a DCP

**Table 20-50. Application Examples for Setting the Transfer Modes of CP A and B of a DCP**

CP A	CP B	
One shot	Not used	Buffer A can be used as a "one shot" buffer. A buffer full interrupt enabled for CP A can signal reaching the end of the buffer.
Auto switch	One shot	Can double the buffer size for a "one shot" buffer. A buffer full interrupt enabled for CP B can signal reaching the end of the buffer.
Circular	Circular	The CPU can switch the buffers at arbitrary times. It will fill or read the frozen buffer during the other buffer is filled or read by the HTU. Interrupts are not required for this case.
Auto switch	Auto switch	Buffer full interrupts (enabled for CP A and B) signal when the end of a buffer is reached. After one buffer is completed the according CPU interrupt routine will read or refill this buffer. At the same time the other buffer is read or filled by the HTU. Here the time when the buffer must be read is determined by the time of the interrupt (i.e. determined by the frequency of the NHET transfer requests).

### 20.6.2 Software Example Sequence Assuming Circular Mode for both CP A and B

The example assumes the NHET address to be read and the main memory address to be written.

- I1 CPU initializes initial DCP: IFADDRA, IFADDRB, IHADDRCT, ITCOUNT
- I2 CPU clears current DCP: CFADDRA, CFADDRB, CFTCTA, CFTCTB
- I3 CPU clears BFINTFL flag of CP A and B
- I4 Enable CP A with the CPENA register. Now the HTU fills buffer A

After some time the CPU intends to read buffer A:

- A1 CPU enables CP B and disables CP A by writing to the CPENA register. After this switch, the HTU fills buffer B. Filling buffer B starts with its initial full address and initial frame counter
- A2 CPU waits for CP A busy bit equals zero
- A3 Optional: CPU verifies that the CP A request lost flag is not set. The bus error flag of CP A could also be checked
- A4 CPU reads the frozen CFTCTA, which indicates the fill level in the buffer
- A5 CPU sets current CP A (CFTCTA and/or CFADDRA) to zero. This allows to find out if any request has happened during the next time buffer A is active
- A6 CPU reads BFINTFL flag of buffer A
- A7 CPU clears the BFINTFL flag of buffer A. This is an initialization for the next time buffer A is used
- A8 CPU reads valid values of frozen buffer A. After reading the CPU does not need to clear the frozen buffer A

After some time the CPU intends to read buffer B:

- B1 CPU enables CP A and disables CP B by writing to the CPENA register. After this switch, the HTU fills buffer A. Filling buffer A starts with its initial full address and initial frame counter
- B2 CPU waits for CP B busy bit equals zero
- B3 Optional: CPU verifies that the CP B request lost flag is not set. The bus error flag of CP B could also be checked
- B4 CPU reads the frozen CFTCTB, which indicates the fill level in the buffer
- B5 CPU sets current CP B (CFTCTB and/or CFADDRB) to zero. This allows to find out if any request has happened during the next time buffer B is active
- B6 CPU reads BFINTFL flag of buffer B
- B7 CPU clears the BFINTFL flag of buffer B. This is an initialization for the next time buffer B is used
- B8 CPU reads valid values of frozen buffer B. After reading the CPU does not need to clear the frozen buffer B

After some time the CPU intends to read buffer A:

A1) ... see above...

---

**NOTE:** The buffer full interrupt doesn't need to be enabled. The BFINTFL flag is used to indicate a circular overrun of the buffer. If the BFINTFL flag is set, also the buffer section after the frozen full address could be read.

---

Steps A3 and B3 in the example sequence above imply that request lost interrupts are disabled. The example below assumes that request lost interrupts are enabled.

Request lost detection with interrupt enabled.

### 20.6.3 Example of an Interrupt Dispatch Flow for a Request Lost Interrupt

- A request lost occurs and the interrupt routine starts.
- Reading INTOFFx.INTYPEx shows that RLOSTFL is the interrupt source.
- Reading INTOFFx.CPOFFx = Ah shows that DCP 5 / CP A has caused the RLOSTFL interrupt. The hardware automatically clears bit (2·5+0) in RLOSTFL.
- Reading RLOSTFL= 84h shows that also another request lost event happened on DCP 1 / CP A [bit (2·1+0)] and on DCP 3 / CP B [bit (2·3+1)] at the same time or after the request lost occurred on DCP 5 / CP A.
- Writing back 84h to RLOSTFL clears bits 2 and 7 and the according pending interrupts.

## **General-Purpose Input/Output (GIO) Module**

---



---

This chapter describes the general-purpose input/output (GIO) module. The GIO module provides the family of devices with input/output (I/O) capability. The I/O pins are bidirectional and bit-programmable. The GIO module also supports external interrupt capability.

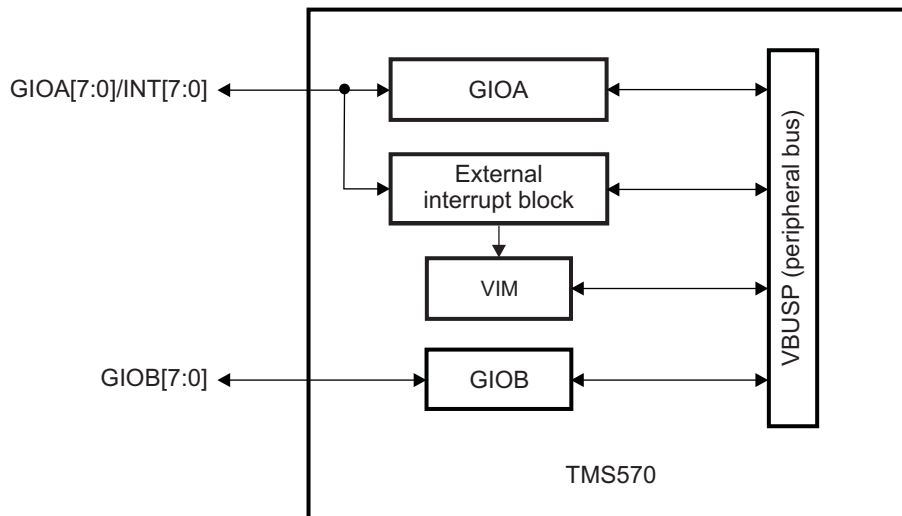
Topic	Page
<b>21.1 Overview</b> .....	<b>1110</b>
<b>21.2 Quick Start Guide</b> .....	<b>1111</b>
<b>21.3 Functional Description of GIO Module</b> .....	<b>1113</b>
<b>21.4 Device Modes of Operation</b> .....	<b>1121</b>
<b>21.5 GIO Control Registers</b> .....	<b>1122</b>
<b>21.6 Applications Examples</b> .....	<b>1143</b>



## 21.1 Overview

The GIO module on this device supports two ports, GIOA and GIOB. Each module has 8 I/O pins. The I/O pins are bidirectional and bit-programmable. The GIOA module also supports external interrupt capability.

**Figure 21-1. Device Level Overview**



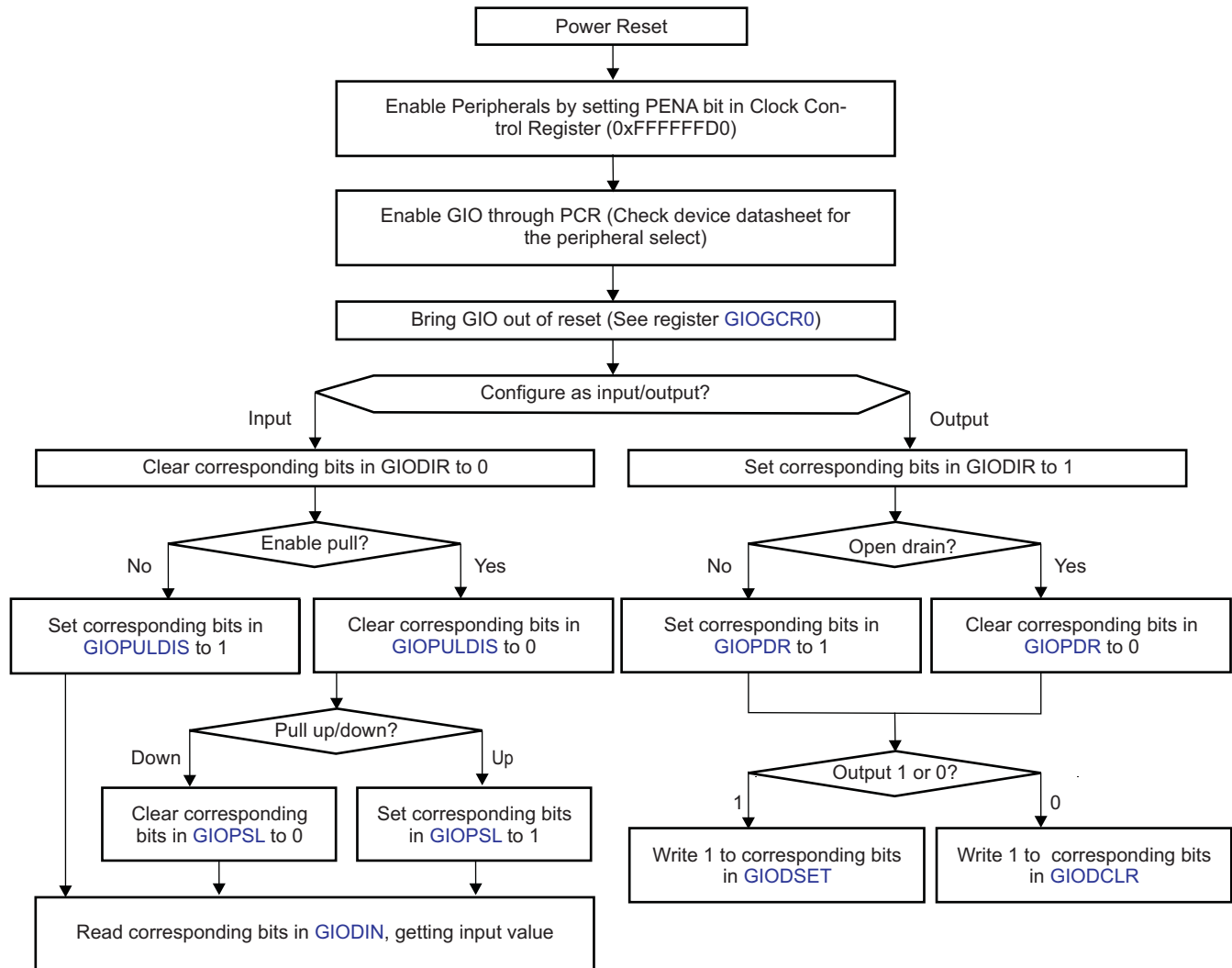
The GIO module has the following features:

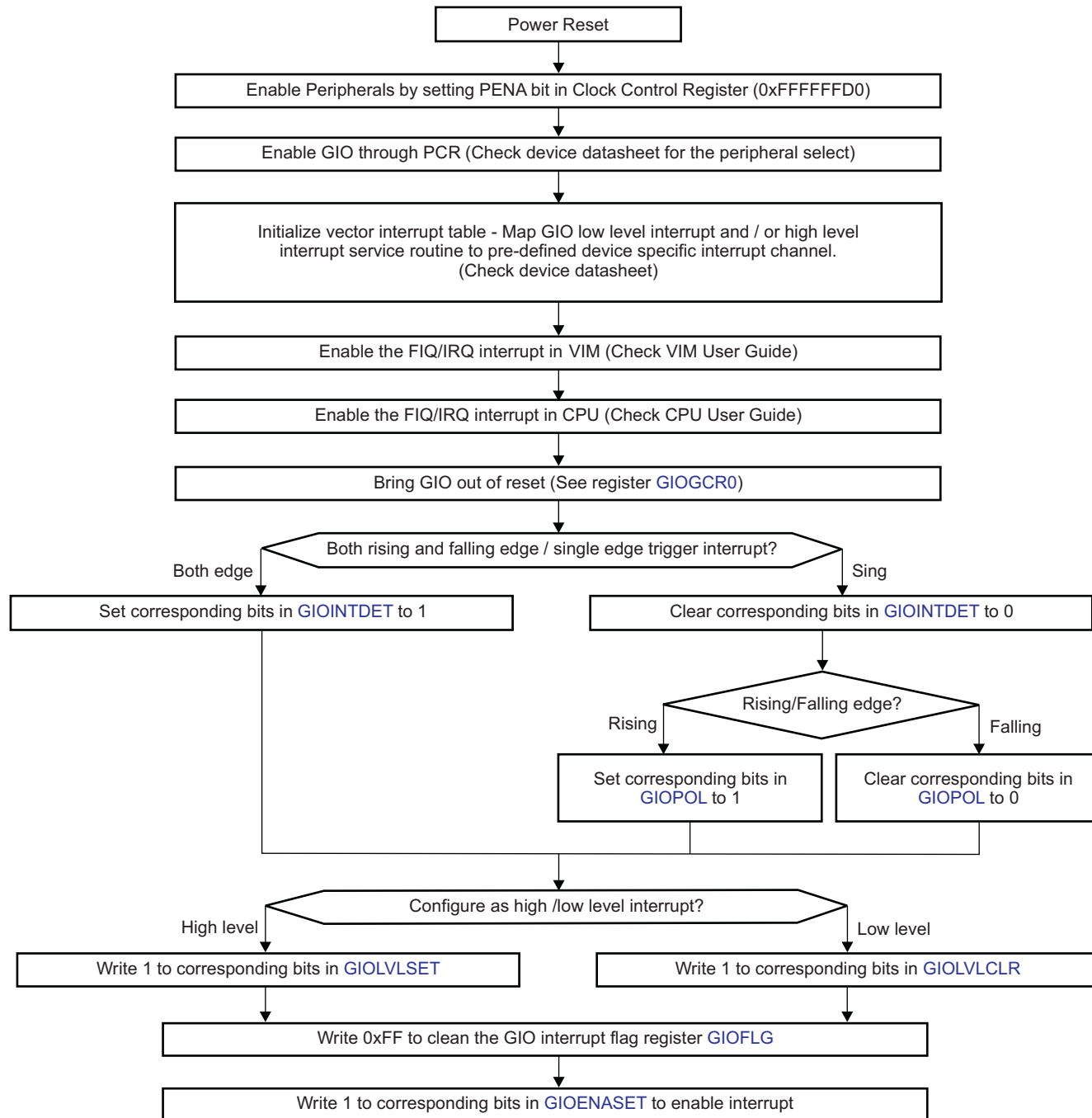
- Each I/O pin is controlled by bits in these registers:
  - Data direction (GIODIR, [Section 21.5.11](#))
  - Data input (GIODIN, [Section 21.5.12](#))
  - Data output (GIODOUT, [Section 21.5.13](#))
  - Data set (GIODSET, [Section 21.5.14](#))
  - Data clear (GIODCLR, [Section 21.5.15](#))
  - Open drain (GIOPDR, [Section 21.5.16](#))
  - Pull disable (GIOPULDIS, [Section 21.5.17](#))
  - Pull select (GIOPSL, [Section 21.5.18](#))
- The interrupts have the following characteristics:
  - Programmable interrupt detection either on both edges or on a single edge (set in GIOINTDET, [Section 21.5.2](#))
  - Programmable edge-detection polarity, either rising or falling edge (set in GIOPOL register, [Section 21.5.3](#))
  - Individual interrupt flags (set in GIOFLG register, [Section 21.5.3](#))
  - Individual interrupt enables, set and cleared through GIOENASET ([Section 21.5.4.1](#)) and GIOENACLR ([Section 21.5.4.2](#)) registers respectively
  - Programmable interrupt priority, set through GIOLVLSET ([Section 21.5.5.1](#)) and GIOLVLCLR ([Section 21.5.5.2](#)) registers
- Internal pullup/pulldown allows unused I/O pins to be left unconnected.

## 21.2 Quick Start Guide

The GIO module comprises two separate components: an input/output (I/O) block and an external interrupt block. [Figure 21-2](#) and [Figure 21-3](#) shows what the user should do after power reset to configure the GIO module as I/O and external interrupt.

**Figure 21-2. I/O Function Quick Start Flow Chart**



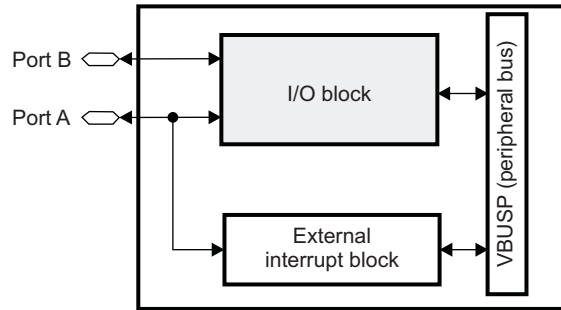
**Figure 21-3. External Interrupt Function Quick Start Flow Chart**

In GIO interrupt service routine, user shall read the GIO offset register GIOOFFA (Section 21.5.7) / GIOOFFB (Section 21.5.8) (depending on high/low level interrupt) to clear the flag and find the pending interrupt GIO channel.

### 21.3 Functional Description of GIO Module

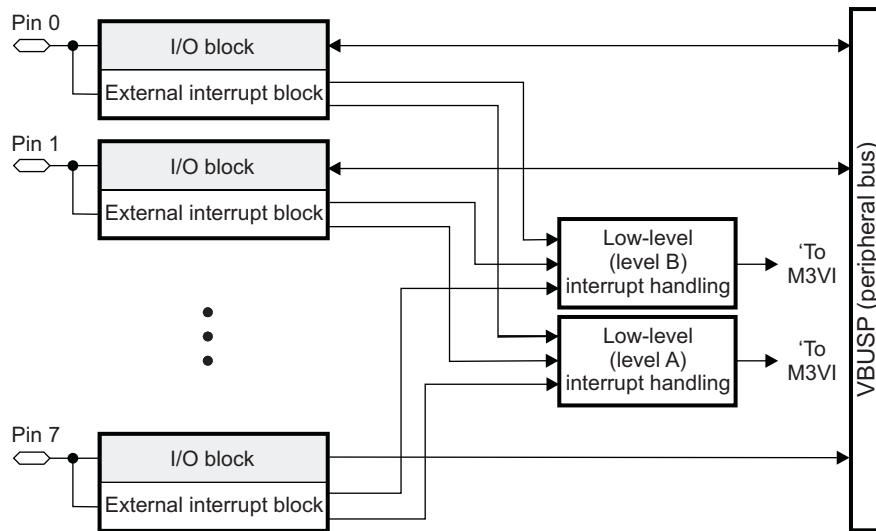
The GIO module (see Figure 21-4) comprises two separate components: an input/output (I/O) block and an external interrupt block. The GIO module has 2 ports (A and B) with 8 pins (0 through 7) per port. The pins for port B handle the standard I/O functions only. Port A can handle interrupts in addition to the standard I/O functions.

Figure 21-4. GIO Module Diagram



The pins on port A, shown in Figure 21-5, 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 (VIM) for processing. The interrupt priority of level A and level B interrupt handling blocks can be re-programmed in the VIM.

Figure 21-5. GIO Port A Module Diagram

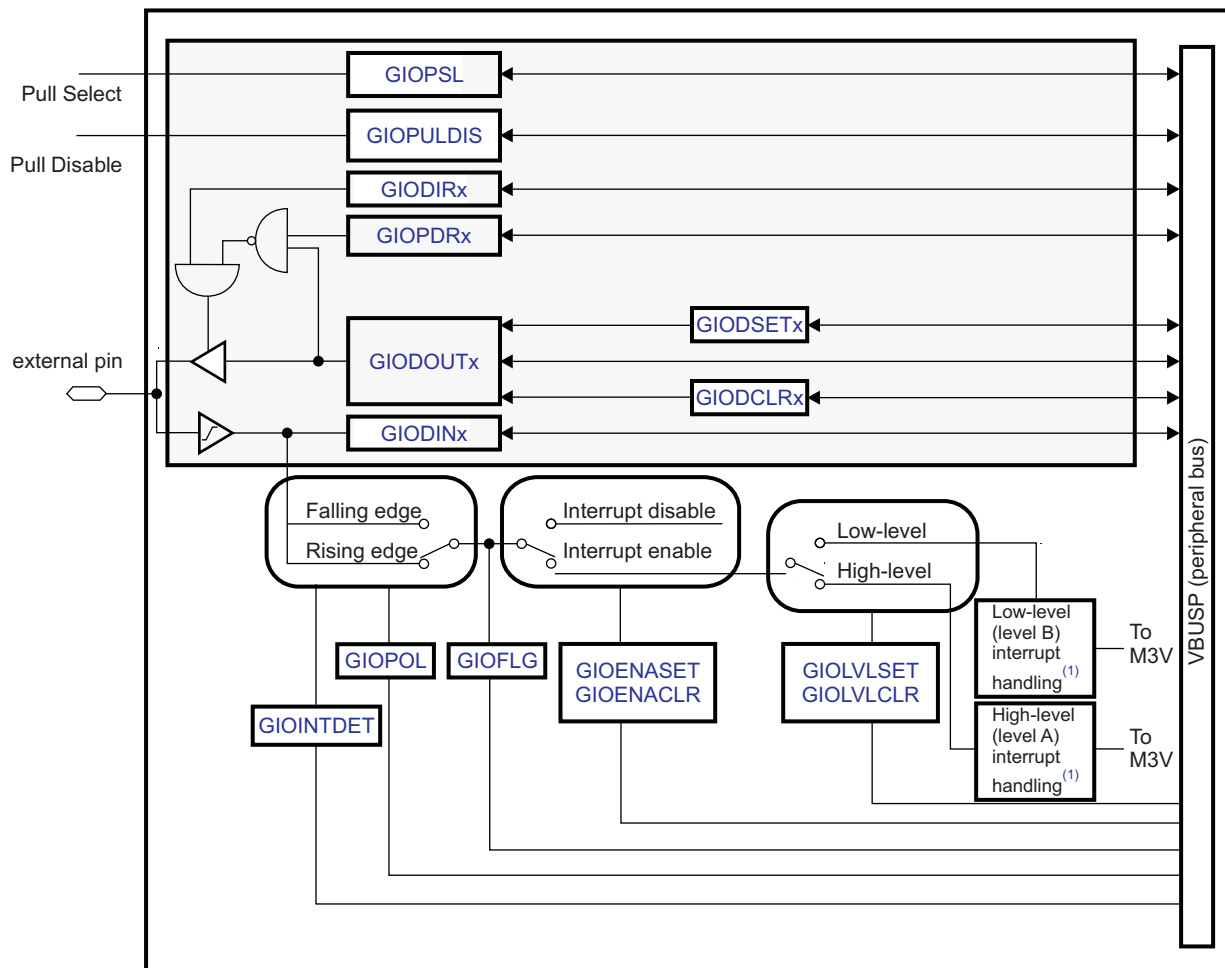


#### 21.3.1 GIO Block Diagram

The GIO block diagram (see Figure 21-6) 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 port B is not interrupt-capable, the block diagram for that port reduces to the shaded portion of the block diagram. The following block diagrams are conditional. The first block diagram is for devices with open drain, pull disable capability, and pullup/pulldown selectability. The second block diagram is for devices with no open drain and no programmable pullup/pulldown (Andromeda).

Figure 21-6. 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.

## 21.3.2 I/O Blocks

Each pin serviced by port A and B contains its individual I/O block.

### 21.3.2.1 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 21-6). The associated registers are:

- GIODIR[7:0] — 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 21-1. When the input buffer is enabled, information that is sent to the external pin is also received in the input buffer. The input buffer is always enabled. Therefore, the information that is sent to the external pin is also received in the input buffer.
- GIODOUT[7:0] — 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 (The input buffer is always enabled. Therefore, the information that is sent to the external pin is also received in the input buffer. 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).

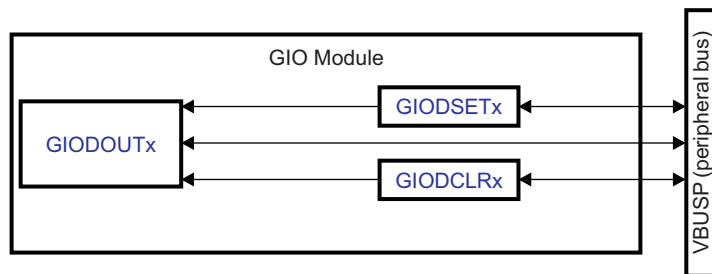
- GIODIN[7:0] — Receives the information from the external pin. A high voltage ( $V_{IH}$  or greater) applied to the pin causes a high value (1) in the data input register ( GIODIN[7:0]). When a low voltage ( $V_{IL}$  or less) is applied to the pin, the data input register reads a low value (0). The  $V_{IH}$  and  $V_{IL}$  values are device specific and can be found in the device datasheet.
- GIOPDR[7:0] — 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] — 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] — Selects the pull type at pin. Pull down or pull up can be selected using the GIOPSL[7:0] register.

### 21.3.2.2 Output Control Registers

When a GIO pin is configured as an output pin, the value in the data output register (GIODOUT[7:0]) specifies the voltage applied to the external pin. The GIO module provides three ways of communicating with the data output register (see Figure 21-7).

- The control register bit can be written directly by writing to the data output register (GIODOUT[7:0]).
- The data output register bit can be set to 1 using the data set register (GIODSET[7:0]).
- The data output register bit can be cleared to 0 using the data clear register (GIODCLR[7:0]).

Figure 21-7. 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.

### 21.3.2.3 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. 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. Programming GIOPSL[7:0] overrides the default pull on the pins. See Table 21-1.

The default pullup/pulldown functionality of all GIO pins at system reset is pulldown.

**NOTE:** It is possible to disable the GIO Input buffer altogether by configuring the pin direction to input via the GIODIR[A-B][7:0], disabling the pin's pull functionality using GIOPULDIS[7:0], and setting the pin's corresponding bit in the GIOPSL[7:0] to 0. Once an input buffer is disabled, the external signal can not be read from the buffer internally.

### 21.3.2.4 Open Drain Function

The GIO pins can be configured to include an open drain functionality using the GIOPDR[7:0] registers, 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] 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.

### 21.3.2.5 Summary

The behavior of the input buffer, output buffer, and the pull control is summarized in [Table 21-1](#).

**Table 21-1. Input Buffer, Output Buffer, and Pull Control Behavior as GPIO Pins <sup>(1)</sup>**

Device under Reset?	Pin Direction (GIODIR) <sup>(2)</sup>	Pull Disable (GIOPULDIS) <sup>(3)</sup>	Pull Select (GIOPSL) <sup>(4)</sup>	Pull Control	Output Buffer <sup>(5)</sup>	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

<sup>(1)</sup> X = Don't care

<sup>(2)</sup> GIODIR = 0 for input, 1 for output

<sup>(3)</sup> GIOPULDIS = 0 for enabling pull control  
= 1 for disabling pull control

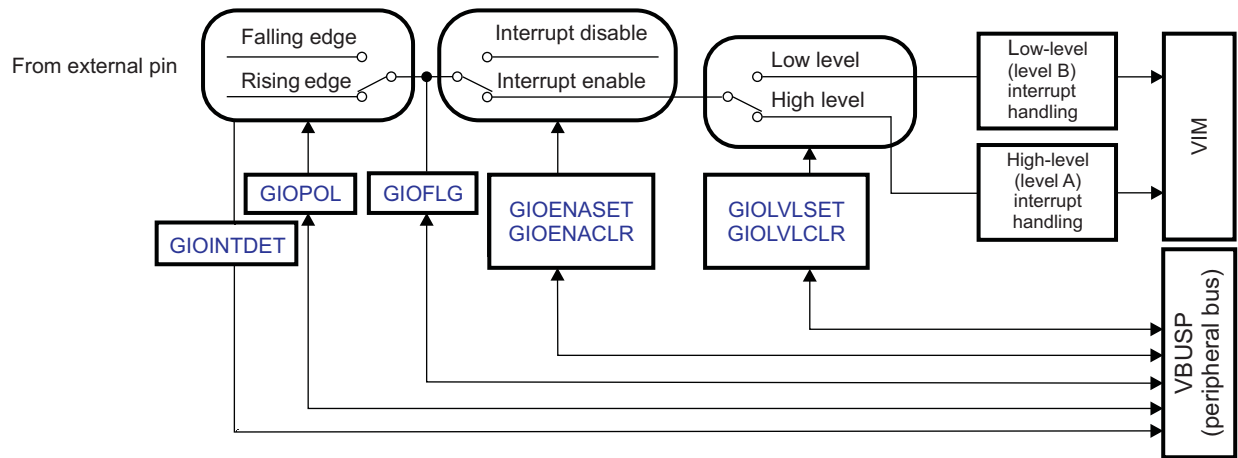
<sup>(4)</sup> GIOPSL = 0 for pull-down functionality  
= 1 for pull-up functionality

<sup>(5)</sup> If open drain is enabled, output buffer will be disabled if output 1.

### 21.3.3 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 (VIM) in the system module (see [Section 21.3.3.3](#)). [Figure 21-8](#) shows the external interrupt block.

Figure 21-8. External Interrupt Block



### 21.3.3.1 Edge Detection and the Flag Register

The edge-detection hardware and flag register, like the input buffer, are always enabled. The GIOINTDET register 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 is used to define whether a rising edge or a falling edge is recognized as an interrupt.

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. (For an explanation of VCLK, please refer to TMS570 architecture user guide).

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 21.4.2 for more information.)

The corresponding flag in the GIOFLG register 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.

### 21.3.3.2 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 21-8).



The external interrupt can be enabled or disabled using the GIOENASET and GIOENACLK register 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 (GIOLVLSSET and GIOLVLCCLR) 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 (VIM), 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 VIM. Refer to the VIM 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. Please refer to VIM user guide for wakeup configuration.

---

Table 21-2 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 (VIM) for servicing because they have the lowest bit values among the high-level and low-level interrupts. Assuming that the VIM 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 VIM 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.

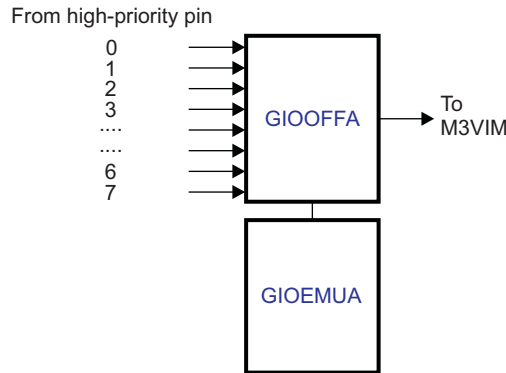
**Table 21-2. Determining Interrupt Priority**

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

### 21.3.3.3 High-Level-Interrupt Block and Low-Level-Interrupt Block

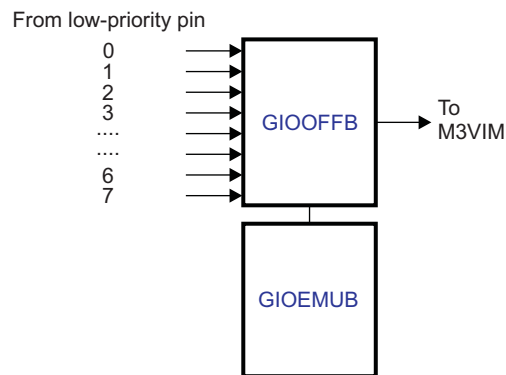
The interrupt-handling blocks each contain two registers: an offset register and an emulation register. The high-level interrupts are denoted as level A and consequently, the registers for the high-level-interrupt-handling block are GIOFFA and GIOEMUA; see Figure 21-9.

**Figure 21-9. High-Level-Interrupt-Handling Block**



Likewise, the registers for the low-level-interrupt-handling block (denoted as level B) are GIOFFB and GIOEMUB; see [Figure 21-10](#).

**Figure 21-10. Low-Level-Interrupt-Handling Block**



The read-only registers GIOFFA and GIOFFB generate a numerical offset value that represents the highest-priority pending external interrupt (see [Table 21-3](#)). 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, GIOFFA, receives signals from each active interrupt that is configured as high-level. The GIOFFA 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 GIOFFB displays the low-level interrupt with the highest priority.

The emulation control registers (GIOEMUA and GIOEMUB) mirror the offset registers. The emulation registers contain a numerical offset value that represent the highest priority pending external interrupt (see [Table 21-3](#)). 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 21-3. GIO Offset A Values and Corresponding Interrupt**

GIOFF(5-0)	Pending External Interrupt
0	No interrupt
1h	Interrupt 0
2h	Interrupt 1
3h	Interrupt 2
4h	Interrupt 3
...	...
8h	Interrupt 7
9h-3Fh	Reserved

Table 21-4 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, GIOFFB, returns a value of 1h, indicating that the interrupt on pin 0 is currently being processed (see Table 21-3).

**Table 21-4. 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
GIOFFA	0	0	0	0	0	0	0	0
GIOEMUA	0	0	0	0	0	0	0	0
GIOFFB	0	0	0	0	0	0	0	1
GIOEMUB	0	0	0	0	0	0	0	1

#### 21.3.3.4 Special Considerations for Interrupts

Please note that interrupts are subject to the following special considerations:

- 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 [Section 21.6.1](#), *Example: Setting Interrupts and Configuring Pins for Output*.
- 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 21.6.3](#), *Example: Clearing Interrupt Flags and Setting Interrupts*.
- 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) F8h. See [Section 21.6.4](#), *Example: Reading Port B Input Register*.

## 21.4 Device Modes of Operation

The GIO module behaves differently in different modes of operation. There are two main modes:

- Emulation mode
- Power-down mode (low-power mode)
  - Module level power-down
  - Device level power-down

### 21.4.1 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 and Emulation Registers**

Emulation mode is a mode of operation of the device and is separate from the GIO emulation registers (GIOEMUA and GIOEMUB). The contents of these emulation registers are identical to the contents of GIO offset registers (GIOOFFA and GIOOFFB). Both emulation registers and GIO offset registers are NOT cleared when they are read in emulation mode. GIO offset registers are cleared when they are read in normal mode (other than emulation mode). The emulation registers are NOT cleared when they are read in normal mode. The intention for the emulation registers is that software can use them without clearing the flags.

---

During emulation mode:

- External interrupts are not captured because the VIM is unable to service interrupts.
- Any register can be read without affecting the state of the system.
- A write to a register affects the state of the system.

### 21.4.2 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 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.

#### 21.4.2.1 Module-Level Power Down

The GIO module can be placed into a power down state by disabling the GIO peripheral module via the appropriate bit in the peripheral power down register. Please refer to the Peripheral Central Resource Registers (PCR Control Registers) for details.

#### 21.4.2.2 Device-Level Power Down

The entire device can be placed in one of the pre-defined low-power modes: doze, snooze, or sleep. See the device datasheet for details.

## 21.5 GIO Control Registers

Table 21-5 shows the summary of the GIO registers. It has two ports designated A-B (GIOA is 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-B][7:0], GIODIN[A-B][7:0], GIODOUT[A-B][7:0], GIODSET[A-B][7:0], GIODCLR[A-B][7:0], GIOPDR[A-B][7:0], GIOPULDIS[A-B][7:0], and GIOPSL[A-B][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 FFF7 BC00h.

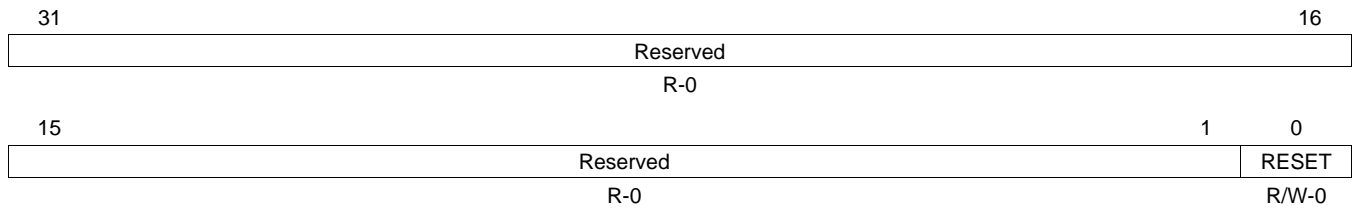
**Table 21-5. GIO Control Registers Summary**

Offset	Acronym	Register Description	Section
FFF7 BC00h	GIOGCR0	GIO Global Control Register	<a href="#">Section 21.5.1</a>
FFF7 BC08h	GIOINTDET	GIO Interrupt Detect Register	<a href="#">Section 21.5.2</a>
FFF7 BC0Ch	GIOPOL	GIO Interrupt Polarity Register	<a href="#">Section 21.5.3</a>
FFF7 BC10h	GIOENASET	GIO Interrupt Enable Set Register	<a href="#">Section 21.5.4.1</a>
FFF7 BC14h	GIOENACLAR	GIO Interrupt Enable Clear Register	<a href="#">Section 21.5.4.2</a>
FFF7 BC18h	GIOLVLSET	GIO Interrupt Priority Set Register	<a href="#">Section 21.5.5.1</a>
FFF7 BC1Ch	GIOLVLCLR	GIO Interrupt Priority Clear Register	<a href="#">Section 21.5.5.2</a>
FFF7 BC20h	GIOFLG	GIO Interrupt Flag Register	<a href="#">Section 21.5.6</a>
FFF7 BC24h	GIOOFFA	GIO Offset A Register	<a href="#">Section 21.5.7</a>
FFF7 BC28h	GIOOFFB	GIO Offset B Register	<a href="#">Section 21.5.8</a>
FFF7 BC2Ch	GIOEMUA	GIO Emulation A Register	<a href="#">Section 21.5.9</a>
FFF7 BC30h	GIOEMUB	GIO Emulation B Register	<a href="#">Section 21.5.10</a>
FFF7 BC34h, FFF7 BC54h	GIODIR[A-B]	GIO Data Direction Register	<a href="#">Section 21.5.11</a>
FFF7 BC38h, FFF7 BC58h	GIODIN[A-B]	GIO Data Input Register	<a href="#">Section 21.5.12</a>
FFF7 BC3Ch, FFF7 BC5Ch	GIODOUT[A-B]	GIO Data Output Register	<a href="#">Section 21.5.13</a>
FFF7 BC40h, FFF7 BC60h	GIODSET[A-B]	GIO Data Set Register	<a href="#">Section 21.5.14</a>
FFF7 BC44h, FFF7 BC64h	GIODCLR[A-B]	GIO Data Clear Register	<a href="#">Section 21.5.15</a>
FFF7 BC48h, FFF7 BC68h	GIOPDR[A-B]	GIO Open Drain Register	<a href="#">Section 21.5.16</a>
FFF7 BC4Ch, FFF7 BC6Ch	GIOPULDIS[A-B]	GIO Pull Disable Register	<a href="#">Section 21.5.17</a>
FFF7 BC50h, FFF7 BC70h	GIOPSL[A-B]	GIO Pull Select Register	<a href="#">Section 21.5.18</a>

**21.5.1 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 21-11](#) and [Table 21-6](#) describe this register.

**Figure 21-11. GIO Global Control Register (GIOGCR0) [offset = FFF7 BC00h]**



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

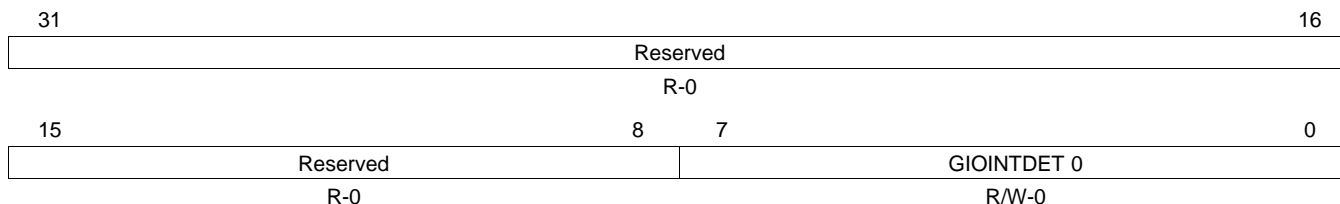
**Table 21-6. GIO Global Control Register (GIOGCR0) Field Descriptions**

Bit	Field	Value	Description
31-1	Reserved	0	Read returns 0. Writes have no effect.
0	RESET	0	The GIO is in reset state.
		1	The GIO is operating normally.

### 21.5.2 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. To ensure recognition of the signal as an edge, the signal must maintain the new level for at least one VCLK cycle. Figure 21-12 and Table 21-7 describe this register.

**Figure 21-12. GIO Interrupt Detect Register (GIOINTDET) [offset = FFF7 BC08h]**



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

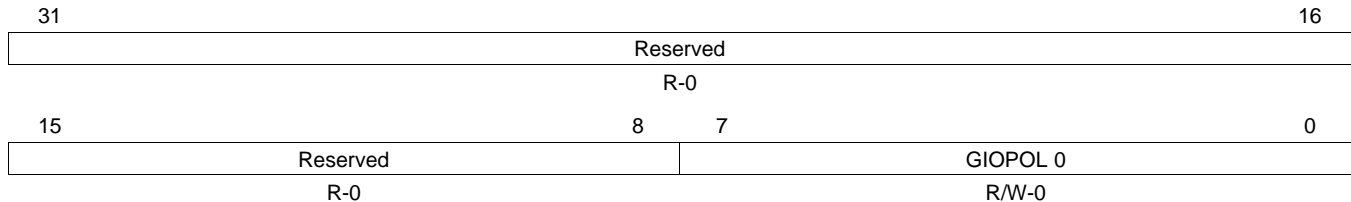
**Table 21-7. GIO Interrupt Detect Register (GIOINTDET) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Read returns 0. Writes have no effect.
7-0	GIOINTDET 0		Interrupt detection select for pins GIOA[7:0]
		0	The flag sets on either a falling or a rising edge on the corresponding pin, depending on the polarity setup in the polarity register ( GIOPOL).
		1	The flag sets on both the rising and falling edges on the corresponding pin.

### 21.5.3 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 21-13](#) and [Table 21-8](#) describe this register.

**Figure 21-13. GIO Interrupt Polarity Register (GIOPOL) [offset = FFF7 BC0Ch]**



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

**Table 21-8. GIO Interrupt Polarity Register (GIOPOL) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Read returns 0. Writes have no effect.
7-0	GIOPOL 0		Interrupt polarity select for pins GIOA[7:0] Normal operation (user or privileged mode): 0 The flag is set on the falling edge on the corresponding pin. 1 The flag is set on the rising edge on the corresponding pin. Low-power mode (doze, snooze, sleep, or hibernate): 0 The interrupt is triggered on the low level. 1 The interrupt is triggered on the high level.



### 21.5.4 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.

#### 21.5.4.1 GIOENASET Register

Figure 21-14 and Table 21-9 describe this register.

---

**NOTE: Enabling Interrupt at the Device Level**

GIO can be mapped to two different device level interrupts, the GIO high level (level A) and low level (level B) interrupts through programming registers GIOLVLSET and GIOLVLCLR. The corresponding bit to the mapped device level interrupt must be set within the vectored interrupt manager (VIM) in the interrupt mask register (REQMASK) to enable the appropriate interrupts. Additionally, the ARM CPU (CPSR bit 7 or 6) must be cleared to recognize interrupt requests (IRQ/FIQ).

---

**Figure 21-14. GIO Interrupt Enable Set Register (GIOENASET) [offset = FFF7 BC10h]**

31	Reserved		16
	R-0		
15	8	7	0
	Reserved		GIOENASET 0
	R-0		R/W-0

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

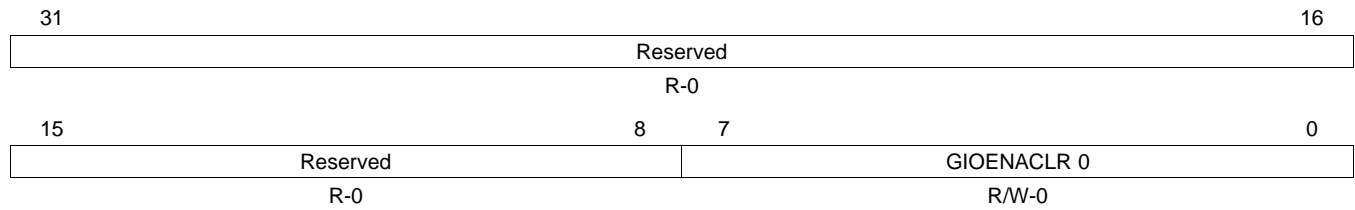
**Table 21-9. GIO Interrupt Enable Set Register (GIOENASET) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Read returns 0. Writes have no effect.
7-0	GIOENASET 0	0	Interrupt enable for pins GIOA[7:0] <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a zero to this bit has no effect.
		1	<i>Read or write:</i> The interrupt is enabled.

### 21.5.4.2 GIOENACLR Register

This register disables the interrupt. [Figure 21-15](#) and [Table 21-10](#) describe this register.

**Figure 21-15. GIO Interrupt Enable Clear Register (GIOENACLR) [offset = FFF7 BC14h]**



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

**Table 21-10. GIO Interrupt Enable Clear Register (GIOENACLR) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Read returns 0. Writes have no effect.
7-0	GIOENACLR 0	0	Interrupt disable for pins GIOA[7:0]. <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a zero to this bit has no effect.
		1	<i>Read:</i> The interrupt is enabled. <i>Write:</i> Writing a one to this bit disables the interrupt.

### 21.5.5 GIO Interrupt Priority Registers (GIOLVLSET and GIOLVLCLR)

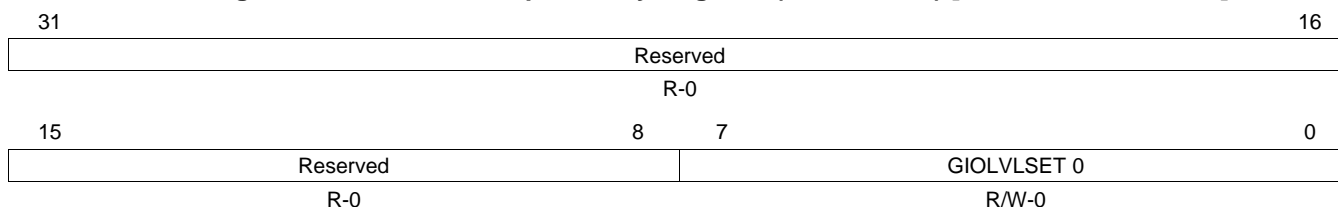
The GIOLVLSET and GIOLVLCLR registers configure the interrupts as high-level (level A) or low-level (level B) going to the VIM. Each interrupt is individually configured.

- The high-level interrupts are recorded to GIOFFA and GIOEMUA.
- The low-level interrupts are recorded to GIOFFB and GIOEMUB.

#### 21.5.5.1 GIOLVLSET Register

The GIOLVLSET register is used to configure an interrupt as a high-level interrupt going to the VIM. An interrupt can be configured as a high level interrupt by writing a 1 into the corresponding bit of the GIOLVLSET register. Writing a zero has no effect. [Figure 21-16](#) and [Table 21-11](#) describe this register.

**Figure 21-16. GIO Interrupt Priority Register (GIOLVLSET) [offset = FFF7 BC18h]**



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

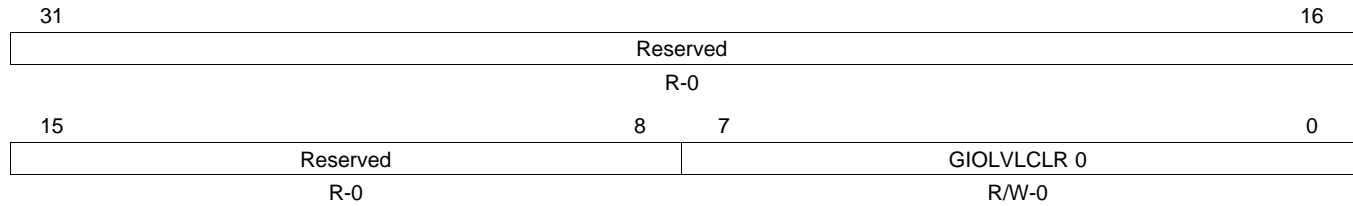
**Table 21-11. GIO Interrupt Priority Register (GIOLVLSET) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Read returns 0. Writes have no effect.
7-0	GIOLVLSET 0	0	GIO high priority interrupt for pins GIOA[7:0]. <i>Read:</i> The interrupt is a low-level interrupt. The low-level interrupts are recorded to GIOFFB and GIOEMUB. <i>Write:</i> Writing a zero to this bit has no effect.
		1	<i>Read or write:</i> The interrupt is set as a high level interrupt. The high-level interrupts are recorded to GIOFFA and GIOEMUA.

### 21.5.5.2 GIOLVLCLR Register

The GIOLVLCLR register is used to configure an interrupt as a low level interrupt going to the VIM. [Figure 21-17](#) and [Table 21-12](#) describe this register.

**Figure 21-17. GIO Interrupt Priority Register (GIOLVLCLR) [offset = FFF7 BC1Ch]**



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

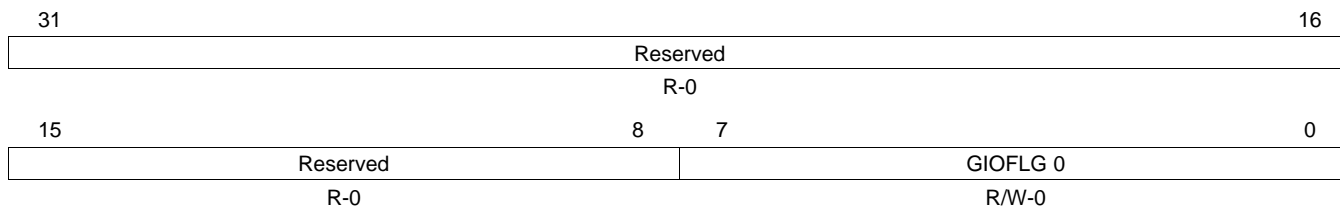
**Table 21-12. GIO Interrupt Priority Register (GIOLVLCLR) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Read returns 0. Writes have no effect.
7-0	GIOLVLCLR 0	0	GIO low priority interrupt for pins GIOA[7:0] <i>Read:</i> The interrupt is a low-level interrupt. <i>Write:</i> Writing a zero to this bit has no effect.
		1	<i>Read:</i> The interrupt is set as a high level interrupt. The high-level interrupts are recorded to GIOFFA and GIOEMUA. <i>Write:</i> The interrupt is set as a low level interrupt. The low-level interrupts are recorded to GIOFFB and GIOEMUB.

### 21.5.6 GIO Interrupt Flag Register (GIOFLG)

The GIOFLG register contains flags indicating that the transition edge (as set in GIOINTDET and GIOPOL) has occurred. The flag is also cleared by reading the appropriate offset register; see Section 21.3.3.3. Figure 21-18 and Table 21-13 describe this register.

**Figure 21-18. GIO Interrupt Flag Register (GIOFLG) [offset = FFF7 BC20h]**



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

**Table 21-13. GIO Interrupt Flag Register (GIOFLG) Field Descriptions**

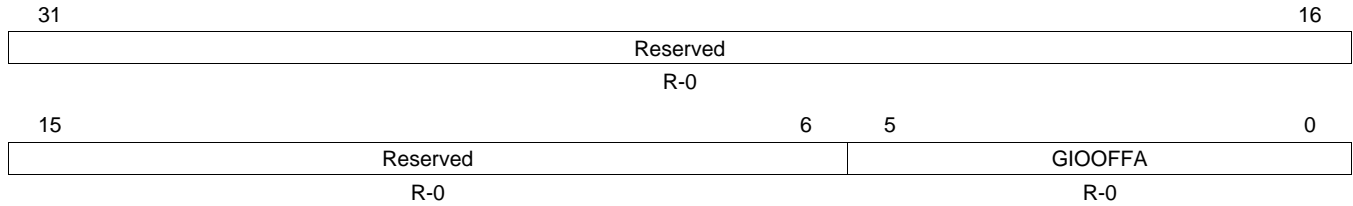
Bit	Field	Value	Description
31-8	Reserved	0	Read returns 0. Writes have no effect.
7-0	GIOFLG 0	0	GIO flag for pins GIOA[7:0]. <i>Read:</i> A transition has not occurred since the last clear. <i>Write:</i> Writing a zero to this bit has no effect.
		1	<i>Read:</i> The selected transition on the corresponding pin has occurred. <i>Write:</i> The corresponding bit is cleared to 0. <b>Note: This bit is also cleared by a read to the corresponding bit in the appropriate offset register</b>

### 21.5.7 GIO Offset A Register (GIOFFA)

The GIOFFA 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 21-19](#) and [Table 21-14](#) describe this register.

**NOTE:** Reading this register clears it, GIOEMUA and the corresponding flag bit in the GIOFLG register. However, in emulation mode, a read to this register does not clear any register or flag.

**Figure 21-19. GIO Offset A Register (GIOFFA) [offset = FFF7 BC24h]**



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

**Table 21-14. GIO Offset A Register (GIOFFA) Field Descriptions**

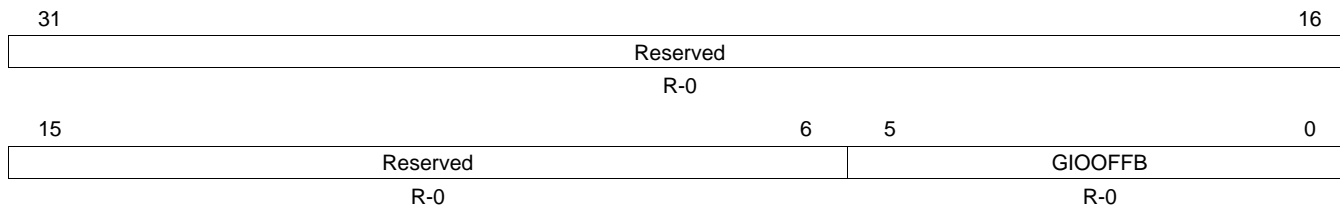
Bit	Field	Value	Description
31-6	Reserved	0	Read returns 0. Writes have no effect.
5-0	GIOFFA	0	GIO offset A. These bits index the currently pending high-priority interrupt. This register and the flag bit (in the GIOFLG register) are also cleared when this register is read, except in emulation mode.
		1h	No interrupt is pending.
		8h	Interrupt 0 (correspond to GIOA0) is pending with a high priority.
		9h-3Fh	Interrupt 7 (correspond to GIOA7) is pending with a high priority.
			Reserved

### 21.5.8 GIO Offset B Register (GIOFFB)

The GIOFFB 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 21-20](#) and [Table 21-15](#) describe this register.

**NOTE:** Reading this register clears it, GIOEMUB and the corresponding flag bit in the GIOFLG register. However, in emulation mode, a read to this register does not clear any register or flag.

**Figure 21-20. GIO Offset B Register (GIOFFB) [offset = FFF7 BC28h]**



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

**Table 21-15. GIO Offset B Register (GIOFFB) Field Descriptions**

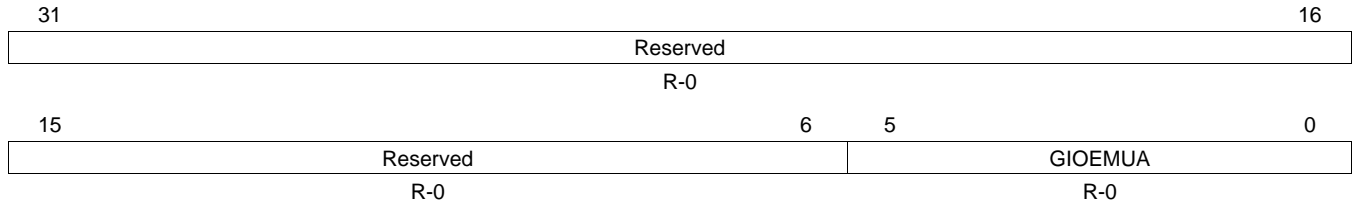
Bit	Field	Value	Description	
31-6	Reserved	0	Read returns 0. Writes have no effect.	
5-0	GIOFFB	0	<b>GIO offset register B.</b> These bits index the currently pending low-priority interrupt. This register and the flag bit (in the GIOFLG register) are also cleared when this register is read, except in emulation mode.	
		1h		Interrupt 0 (correspond to GIOA0) is pending with a low priority.
		8h		Interrupt 7 (correspond to GIOA7) is pending is pending with a low priority.
		9h-3Fh		Reserved

### 21.5.9 GIO Emulation A Register (GIOEMUA)

The GIOEMUA register is a read-only register. The contents of this register are identical to the contents of GIOOFFA. The intention for this register is that software can use it without clearing the flags. Figure 21-21 and Table 21-16 describe this register.

**NOTE:** The corresponding flag in the GIOFLG register is not cleared when the GIOEMUA register is read.

**Figure 21-21. GIO Emulation A Register (GIOEMUA) [offset = FFF7 BC2Ch]**



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

**Table 21-16. GIO Emulation A Register (GIOEMUA) Field Descriptions**

Bit	Field	Value	Description
31-6	Reserved	0	Read returns 0. Writes have no effect.
5-0	GIOEMUA	0	GIO Emulation Register A. These bits index the currently pending high-priority interrupt. No interrupt is pending.
		1h	Interrupt 0 (correspond to GIOA0) is pending with a high priority.
		8h	Interrupt 7 (correspond to GIOA7) is pending with a high priority.
		9h-3Fh	Reserved

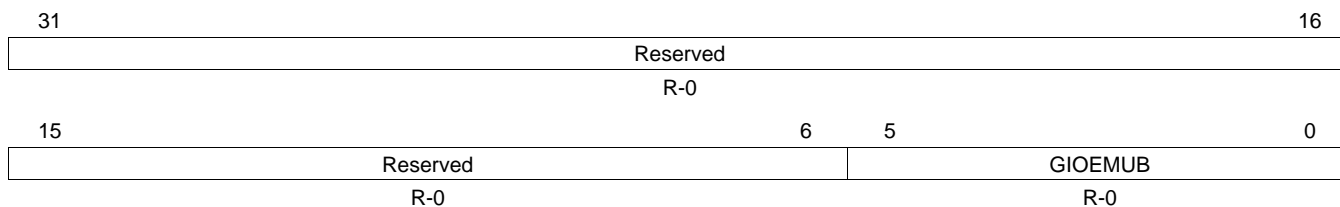


### 21.5.10 GIO Emulation B Register (GIOEMUB)

The GIOEMUB register is a read-only register. The contents of this register are identical to the contents of GIOOFFB. The intention for this register is that software can use it without clearing the flags. Figure 21-22 and Table 21-17 describe this register.

**NOTE:** The corresponding flag in the GIOFLG register is not cleared when the GIOEMUB register is read.

**Figure 21-22. GIO Emulation B Register (GIOEMUB) [offset = FFF7 BC30h]**



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

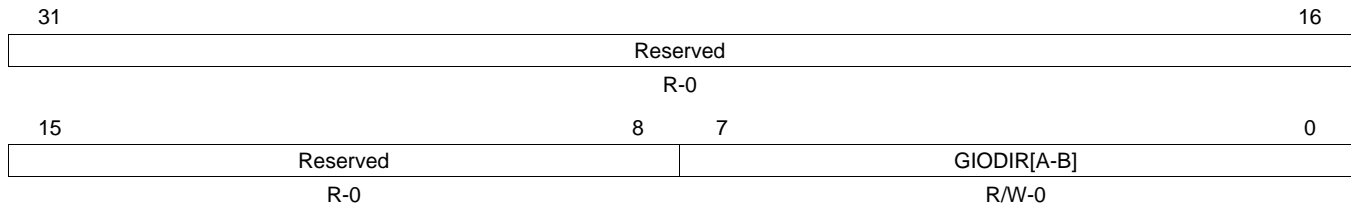
**Table 21-17. GIO Emulation B Register (GIOEMUB) Field Descriptions**

Bit	Field	Value	Description
31-6	Reserved	0	Read returns 0. Writes have no effect.
5-0	GIOEMUB	0	GIO Emulation Register B. These bits index the currently pending low-priority interrupt. No interrupt is pending.
		1h	Interrupt 0 (correspond to GIOA0) is pending with a low priority.
		8h	Interrupt 7 (correspond to GIOA7) is pending with a low priority.
		9h-3Fh	Reserved

### 21.5.11 GIO Data Direction Registers [A-B][7:0] (GIODIR[A-B][7:0])

The GIODIR register controls whether the pins of a given port are configured as inputs or outputs. Figure 21-23 and Table 21-18 describe this register.

**Figure 21-23. GIO Data Direction Registers [A-B][7:0] (GIODIR[A-B]) [offset = FFF7 BC34h, FFF7 BC54h]**



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

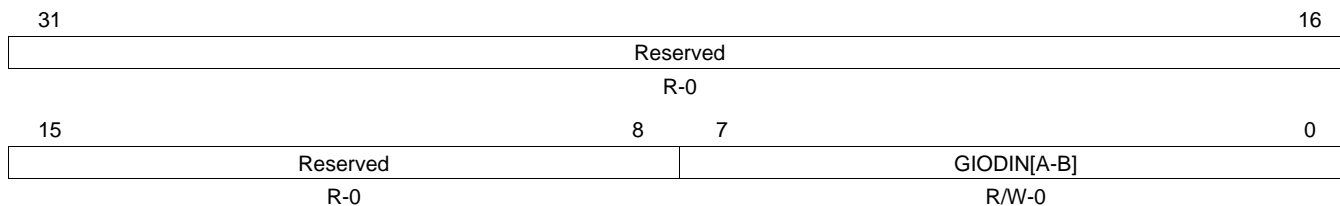
**Table 21-18. GIO Data Direction Registers [A-B][7:0] (GIODIR[A-B][7:0]) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Read returns 0. Writes have no effect.
7-0	GIODIR[A-B]	0	The GIO pin is an input. Note: If the pin direction is set as an input , the output buffer is tristated.
		1	The GIO pin is an output. Note: The input buffer is always enabled except for the situation stated in Pullup/Pulldown Function.

### 21.5.12 GIO Data Input Registers [A-B][7:0] (GIODIN[A-B][7:0])

Values in the GIODIN register reflect the current state (high = 1 or low = 0) on the pins of the port. [Figure 21-24](#) and [Table 21-19](#) describe this register.

**Figure 21-24. GIO Data Input Registers [A-B][7:0] (GIODIN[A-B]) [offset = FFF7 BC38h, FFF7 BC58h]**



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

**Table 21-19. GIO Data Input Registers [A-B][7:0] (GIODIN[A-B][7:0]) Field Descriptions**

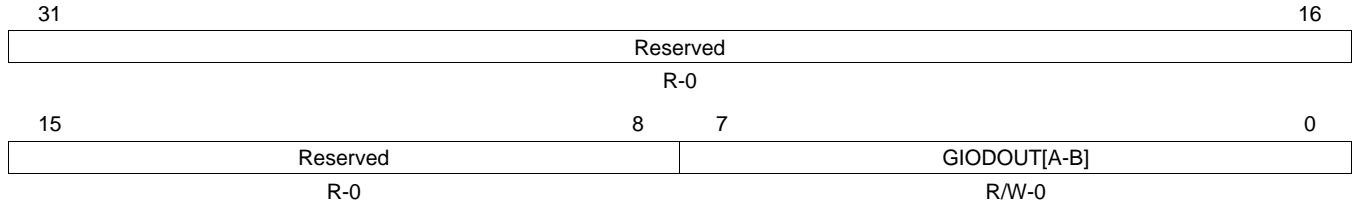
Bit	Field	Value	Description
31-8	Reserved	0	Read returns 0. Writes have no effect.
7-0	GIODIN[A-B]	0	The pin is at logic low (0).
		1	The pin is at logic high (1).

**21.5.13 GIO Data Output Registers [A-B][7:0] (GIODOUT[A-B][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 21-25](#) and [Table 21-20](#) describe this register.

**NOTE:** Values in the GIODSET register, GIO Data Set Register [A-, set the data output control register bits to 1 regardless of the current value in the GIODOUT bits.

**Figure 21-25. GIO Data Output Registers [A-B][7:0] (GIODOUT[A-B][7:0]) [offset = FFF7 BC3Ch, FFF7BC5Ch]**



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

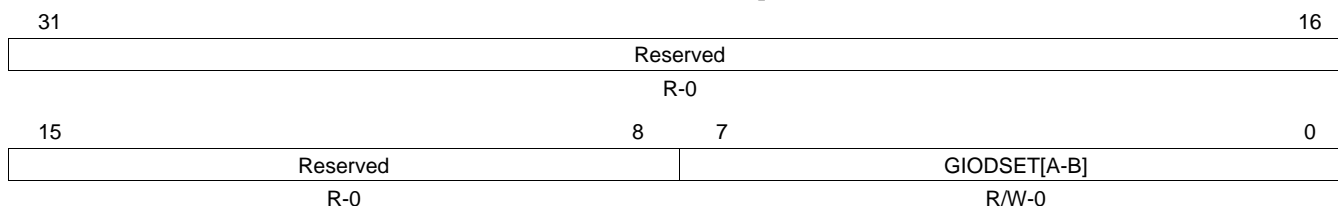
**Table 21-20. GIO Data Output Registers [A-B][7:0] (GIODOUT[A-B][7:0]) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Read returns 0. Writes have no effect.
7-0	GIODOUT[A-B]	0	GIO data output of ports [A-B], pins[7:0]. The pin is driven to logic low (0).
		1	The pin is driven to logic high (1).
			<b>Note: Output is in high impedance state if the GIOPDRx bit = 1 and GIODOUTx bit = 1.</b>
			<b>Note: GIO pin is placed in output mode by setting the GIODIRx bit to 1.</b>

### 21.5.14 GIO Data Set Register [A-B][7:0] (GIODSET[A-B][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 21-26](#) and [Table 21-21](#) describe this register.

**Figure 21-26. GIO Data Set Registers [A-B][7:0] (GIODSET[A-B][7:0]) [offset = FFF7 BC40h, FFF7 BC60h]**



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

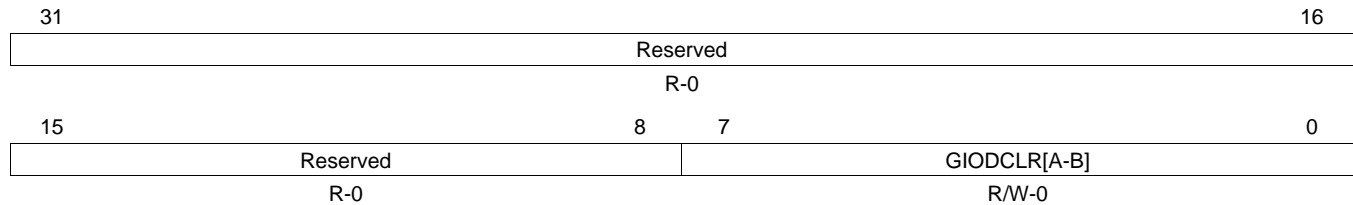
**Table 21-21. GIO Data Set Registers [A-B][7:0] (GIODSET[A-B][7:0]) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Read returns 0. Writes have no effect.
7-0	GIODSET[A-B]	0	GIO data set for ports [A-B], pins[7:0]. This bit drives the output of GIO pin high. <i>Write:</i> Writing a zero has no effect. 1 <i>Write:</i> The corresponding GIO pin is driven to logic high (1). <b>Note:</b> The current logic state of the GIODOUT bit will also be displayed by this bit. <b>Note:</b> GIO pin is placed in output mode by setting the GIODIRx bit to 1.

**21.5.15 GIO Data Clear Registers [A-B][7:0] (GIODCLR[A-B][7:0])**

Values in this register clear the data output register (GIO Data Set Register [A-B]) bit to 0 regardless of its current value. The contents of this register reflect the contents of GIODOUT. [Figure 21-27](#) and [Table 21-22](#) describe this register.

**Figure 21-27. GIO Data Clear Registers [A-B][7:0] (GIODCLR[A-B][7:0]) [offset = FFF7 BC44h, FFF7 BC64h]**



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

**Table 21-22. GIO Data Clear Registers [A-B][7:0] (GIODCLR[A-B][7:0]) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Read returns 0. Writes have no effect.
7-0	GIODCLR[A-B]	0	GIO data clear for ports [A-B], pins[7:0]. This bit drives the output of GIO pin low. <i>Write:</i> Writing a zero has no effect. 1 <i>Write:</i> The corresponding GIO pin is driven to logic low (0). <b>Note:</b> The current logic state of the GIODOUT bit will also be displayed by this bit. <b>Note:</b> GIO pin is placed in output mode by setting the GIODIRx bit to 1.
		1	

### 21.5.16 GIO Open Drain Register [A-B][7:0] (GIOPDR[A-B][7:0])

Values in this register enable or disable the open drain capability of the data pins. [Figure 21-28](#) and [Table 21-23](#) describe this register.

**Figure 21-28. GIO Open Drain Registers [A-B][7:0] (GIOPDR[A-B][7:0]) [offset = FFF7 BC48h, FFF7 BC68h]**

31	Reserved		16
R-0			
15	8	7	0
Reserved		GIOPDR[A-B]	
R-0		R/W-0	

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

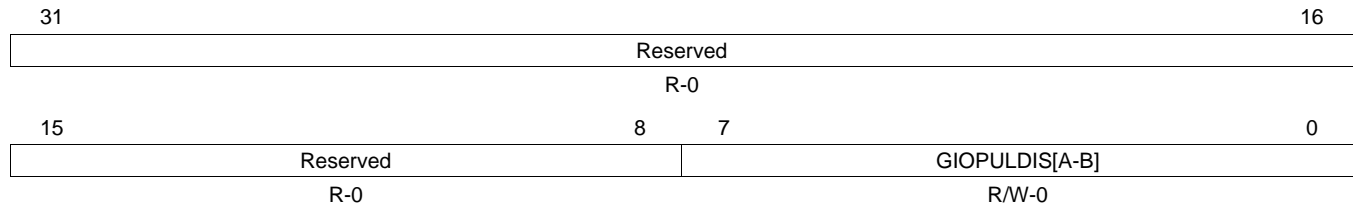
**Table 21-23. GIO Open Drain Registers [A-B][7:0] (GIOPDR[A-B][7:0]) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Read returns 0. Writes have no effect.
7-0	GIOPDR[A-B]	0	The GIO pin is configured in push/pull (normal GIO) mode. The output voltage is $V_{OL}$ or lower if GIODOUT bit = 0 and $V_{OH}$ or higher if GIODOUT bit = 1.
		1	The GIO pin is configured in open drain mode. The GIODOUTx bit controls the state of the GIO output buffer: GIODOUTx = 0 The GIO output buffer is driven low; GIODOUTx = 1 The GIO output buffer is tristated.

**21.5.17 GIO Pull Disable Registers [A-B][7:0] (GIOPULDIS[A-B][7:0])**

Values in this register enable or disable the pull control capability of the pins. [Figure 21-29](#) and [Table 21-24](#) describe this register.

**Figure 21-29. GIO Pull Disable Registers [A-B][7:0] (GIOPULDIS[A-B][7:0]) [offset = FFF7 BC4Ch, FFF7 BC6Ch]**



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

**Table 21-24. GIO Pull Disable Registers [A-B][7:0] (GIOPULDIS[A-B][7:0]) Field Descriptions**

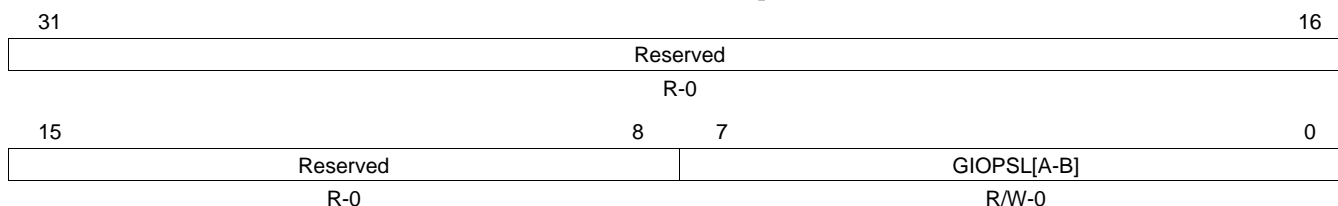
Bit	Field	Value	Description
31-8	Reserved	0	Read returns 0. Writes have no effect.
7-0	GIOPULDIS[A-B]	0	The pull functionality is enabled.
		1	The pull functionality is disabled.
			<b>Note: The GIO pin is placed in input mode by clearing the GIODIRx bit to 0.</b>



### 21.5.18 GIO Pull Select Register [A-B][7:0] (GIOPSL[A-B][7:0])

Values in this register select the pull up or pull down functionality of the pins. [Figure 21-30](#) and [Table 21-25](#) describe this register. Values in this register will disable or enable the input buffer when the pull logic is disabled.

**Figure 21-30. GIO Pull Select Registers [A-B][7:0] (GIOPSL[A-B][7:0]) [offset = FFF7 BC50h, FFF7 BC70h]**



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

**Table 21-25. GIO Pull Select Registers [A-B][7:0] (GIOPSL[A-B][7:0]) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Read returns 0. Writes have no effect.
7-0	GIOPSL[A-B]	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.
			<b>Note: The pull up/pull down functionality is enabled by clearing corresponding bit in GIOPULDIS to 0.</b>

## 21.6 Applications Examples

The application examples in this section assume a typical configuration of five I/O functional ports, in which only Port A is external interrupt-capable.

### 21.6.1 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 0xFFFF7BC00      ;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

```

### 21.6.2 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  0xFFFF7BC00      ;device specific address for the GIO registers.
GIOENA   .equ   0x04             ;setting up equate statements
...
GDIRB   .equ   0x58
GDIRB   .equ   0x5C
GDIRB   .equ   0x60
GDIRB   .equ   0x64
GDIRB   .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, #GDIRB                  ; loads offset address of GDIRB.
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, #GIODCLR                ; loads GIODCLR 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

```

### 21.6.3 Example: Clearing Interrupt Flags and Setting Interrupts

The following code demonstrates how interrupt flags should be cleared before interrupts are enabled. In this example, pins GPIO(2:0) are set as interrupts.

```

GIO_LOC      .word    0xFFFF7BC00      ;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.

```

### 21.6.4 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 0xFFFF7BC00           ;device specific address for GIO registers
GIOENA .equ 0x04                    ;setting up equate statements
...
GIODIRB .equ 0x58
GIODINB .equ 0x5C
GIODOUTB.equ 0x60
GIOSETB.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.

```

## **Controller Area Network (DCAN) Module**

This chapter describes the controller are network (DCAN) module.

Topic	Page
<b>22.1 Overview</b> .....	<b>1148</b>
<b>22.2 CAN Modules</b> .....	<b>1150</b>
<b>22.3 CAN Blocks</b> .....	<b>1151</b>
<b>22.4 CAN Bit Timing</b> .....	<b>1153</b>
<b>22.5 CAN Module Configuration</b> .....	<b>1158</b>
<b>22.6 Message RAM</b> .....	<b>1161</b>
<b>22.7 Message Interface Register Sets</b> .....	<b>1166</b>
<b>22.8 Message Object Configurations</b> .....	<b>1169</b>
<b>22.9 Message Handling</b> .....	<b>1171</b>
<b>22.10 CAN Message Transfer</b> .....	<b>1176</b>
<b>22.11 Interrupt Functionality</b> .....	<b>1178</b>
<b>22.12 Global Power Down Mode</b> .....	<b>1180</b>
<b>22.13 Local Power Down Mode</b> .....	<b>1181</b>
<b>22.14 GIO support</b> .....	<b>1183</b>
<b>22.15 Test Modes</b> .....	<b>1184</b>
<b>22.16 Parity Check Mechanism</b> .....	<b>1188</b>
<b>22.17 Debug/Suspend Mode</b> .....	<b>1189</b>
<b>22.18 DCAN Control Registers</b> .....	<b>1190</b>

## 22.1 Overview

This reference guide gives a general description of the DCAN Controller Area Network module and explains its basic functionality.

The Controller Area Network is a high integrity, serial, multi-master communication protocol for distributed real-time applications. This CAN module is implemented according to ISO 11898-1 and is suitable for industrial, automotive and general embedded communications.

### 22.1.1 Features

The DCAN module provides the following features:

#### Protocol

- Supports CAN protocol version 2.0 part A

#### Speed

- Bit rates up to 1 MBit/s

#### MailBox

- Configurable Message objects
- Individual identifier masks for each message object
- Programmable FIFO mode for message objects

#### High Speed MailBox Access

- DMA access to Message RAM.

#### Power

- Global power down and wakeup support
- Local power down and wakeup support

#### Debug

- Suspend mode for debug support
- Programmable loop-back modes for self-test operation
- Direct access to Message RAM in test mode
- Supports Two interrupt lines - Level 0 and Level 1

#### Others

- Automatic Message RAM initialization
- Automatic bus on after Bus-Off state by a programmable 32-bit timer
- CAN Rx / Tx pins configurable as general purpose IO pins
- Software module reset
- Message RAM parity check mechanism
- Dual clock source to reduce jitter

### 22.1.2 Functional Description

The CAN protocol is an ISO standard (ISO 11898) for serial data communication. This protocol uses Non-Return To Zero (NRZ) with bit-stuffing. And the communication is carried over a two-wire balanced signaling scheme.

The TMS570 DCAN data communication happens through the CAN\_TX and CAN\_RX pins. An additional transceiver hardware is required for the connection to the physical layer (CAN bus) CAN\_High and CAN\_Low.

The DCAN register set can be accessed directly by the CPU. These registers are used to control and configure the CAN module and the Message RAM.

Individual CAN message objects should be configured for communication over a CAN network. The message objects and identifier masks are stored in the Message RAM.

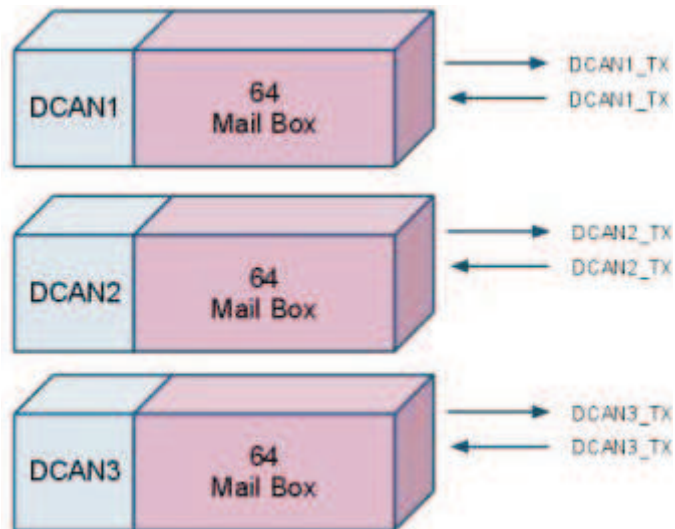
The CAN module internally handles functions such acceptance filtering, transfer of messages from and to the Message RAM, handling of transmission requests as well as the generation of interrupts or DMA requests.



## 22.2 CAN Modules

This device supports three DCAN modules , referred to as DCAN1,DCAN2 and DCAN3 in this document. Each DCAN has its own Message RAM supporting 64 message objects.

**Figure 22-1. DCAN Modules Message Ram**



The base address for DCAN control register and the correspond RAM are as follows:

**DCAN1** base address is **0xFFF7\_DC00** and DCAN1 RAM base address is **0xFF1E\_0000**

**DCAN2** base address is **0xFFF7\_DE00** and DCAN2 RAM base address is **0xFF1C\_0000**

**DCAN3** base address is **0xFFF7\_E000** and DCAN3 RAM base address is **0xFF1A\_0000**

## 22.3 CAN Blocks

The DCAN Module comprises of the following basic blocks.

### 22.3.1 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.

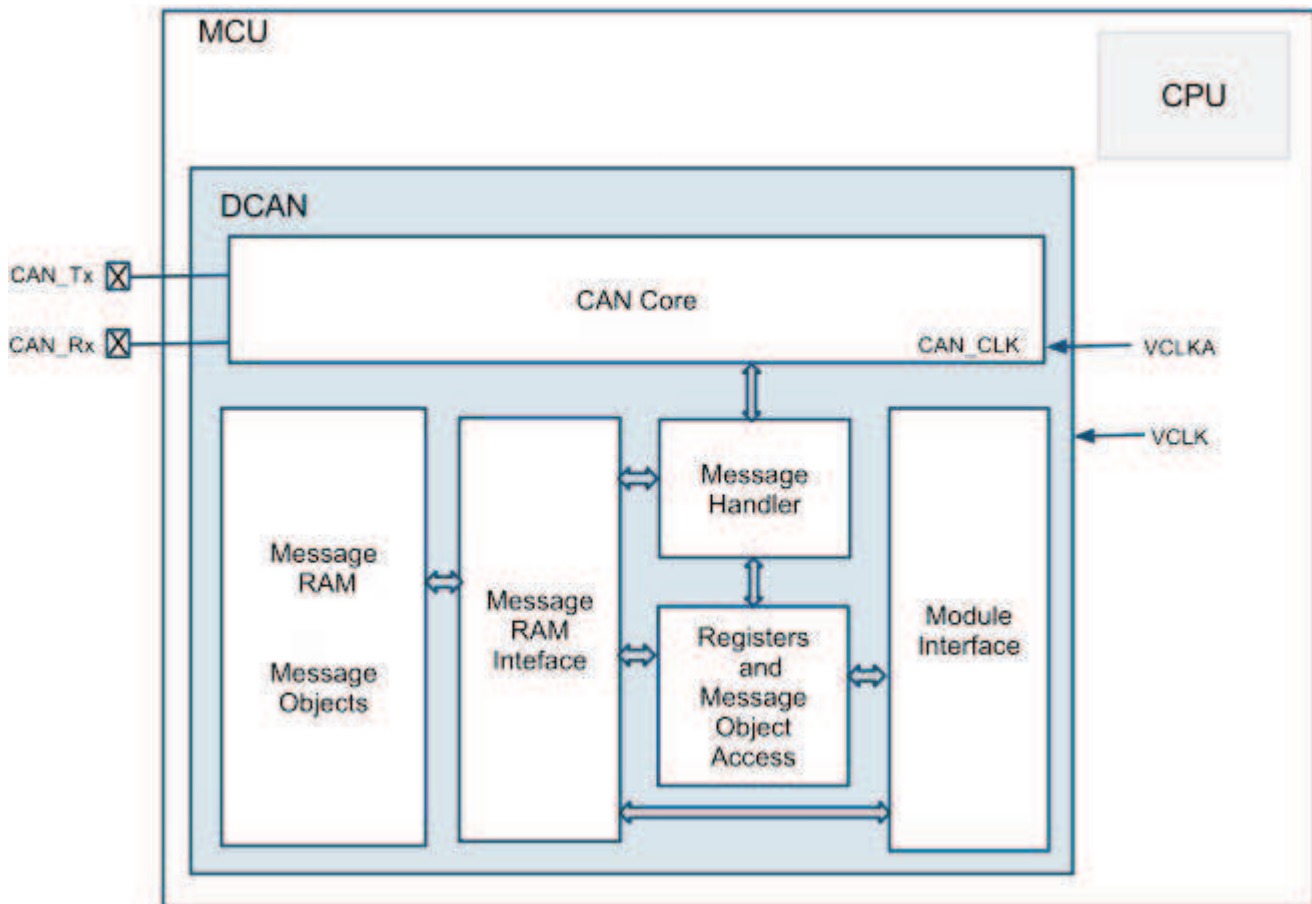
### 22.3.2 Message RAM

The DCAN Message RAM enables storage of CAN messages.

### 22.3.3 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.

Figure 22-2. Block Diagram



### 22.3.4 Message RAM Interface

The Interface Register sets control the CPU read and write accesses to the Message RAM.

There are three interface registers IF1,IF2 and IF3.

IF1 and IF2 Interface Registers sets for read and write access.

IF3 Interface Register set for read access only.

The Interface Registers have the same word-length as the Message RAM.

Additional information can be found in [Section 22.7](#).

### 22.3.5 Register and Message Object Access

During normal operation, data consistency of the message objects is guaranteed by indirectly accessing the message objects through the interface registers IF1 and IF2.

In order to be able to perform tests on the message object memory, a dedicated test mode has been implemented, which allows direct access by either the CPU or DMA. During normal operation direct access has to be avoided.

### 22.3.6 Dual Clock Source

Two clock domains are provided to the DCAN module:

1. VCLK - The peripheral synchronous clock domain as the general module clock source.
2. VCLKA - The peripheral asynchronous clock source domain provided to the CAN core as clock source (CAN\_CLK) for generating the CAN Bit Timing.

If a frequency modulated clock output from FMPLL is used as the VCLK source, then VCLKA should be derived from an unmodulated clock source (e.g. OSCIN source).

The clock source for VCLKA is selected by the Peripheral Asynchronous Clock Source Register in the system module (Additional information can be found in Architecture Specification).

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 refer to the system module reference guide and the device datasheet for more information how to configure the relevant clock source registers in the system module.

Between the two clock domains, a synchronization mechanism is implemented in the DCAN module in order to ensure correct data transfer.

---

**NOTE:** If the dual clock functionality is used, then VCLK must always be higher or equal to CAN\_CLK (derived from the asynchronous clock source), in order to achieve a stable functionality of the DCAN. Here also the frequency shift of the modulated VCLK has to be considered:

$$f_{0, \text{VCLK}} \pm \Delta f_{\text{FM,VCLK}} \geq f_{\text{CANCLK}}$$


---

**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.

---

## 22.4 CAN Bit Timing

The DCAN supports bit rates between less than 1 kBit/s and 1000 kBit/s.

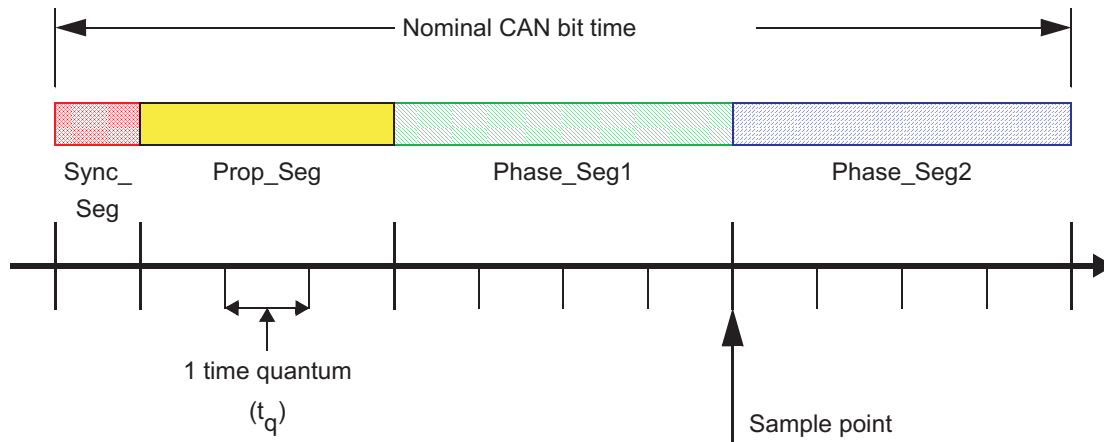
Each member of the CAN network has its own clock generator, typically derived from a crystal oscillator. The Bit timing parameters can be configured individually for each CAN node, creating a common Bit rate even though the CAN nodes' oscillator periods ( $f_{osc}$ ) may be different.

### 22.4.1 Bit Time and Bit Rate

According to the CAN specification, the Bit time is divided into four segments (see Figure 22-3):

- Synchronization Segment (Sync\_Seg)
- Propagation Time Segment (Prop\_Seg)
- Phase Buffer Segment 1 (Phase\_Seg1)
- Phase Buffer Segment 2 (Phase\_Seg2)

Figure 22-3. 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:

$$t_q = \text{Baud Rate Prescaler} / \text{CAN\_CLK}$$

Apart from the fixed length of the synchronization segment, these numbers are programmable.

The Table 22-1 describes the minimum programmable ranges required by the CAN protocol.

A given bit rate may be met by different Bit time configurations.

Table 22-1. 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.

---

### 22.4.1.1 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.

### 22.4.1.2 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.

### 22.4.1.3 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.

### 22.4.1.4 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:

$$(1-df) \times f_{nom} \leq f_{osc} \leq (1+df) \times f_{nom} \quad (31)$$

depends on the proportions of Phase\_Seg1, Phase\_Seg2, SJW, and the bit time. The maximum tolerance  $df$  is defined by two conditions (both shall be met):

$$I: df \leq \frac{\min(TSeg1, TSeg2)}{2 \times (13 \times \text{bit\_time} - TSeg2)}$$

$$II: df \leq \frac{SJW}{20 \times \text{bit\_time}} \quad (32)$$

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  $\mu$ s) with a bus length of 40 m.

### 22.4.2 DCAN Bit Timing Registers

In the DCAN, the bit timing configuration is programmed in two register bytes, additionally a third byte for a baud rate prescaler extension of 4 bits (BREP) is provided. The sum of Prop\_Seg and Phase\_Seg1 (as TSEG1) is combined with Phase\_Seg2 (as TSEG2) in one byte, SJW and BRP (plus BRPE in third byte) are combined in the other byte

In this bit timing register, the components TSEG1, TSEG2, SJW and BRP have to be programmed to a numerical value that is one less than its functional value; so instead of values in the range of [1... n], values in the range of [0 ... n-1] are programmed. That way, 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 (BTR) is the configuration input of the CAN protocol controller. The Baud Rate Prescaler (configured by BRPE/BRP) defines the length of the time quantum (the basic time unit of the bit time); the Bit Timing Logic (configured by TSEG1, TSEG2, and SJW) defines the number of time quanta in the bit time.

#### 22.4.2.1 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  $t_q = (\text{Baud Rate Prescaler}) / \text{CAN\_CLK}$ . Several combinations may lead to the desired bit time, allowing iterations of the following steps.

First part of the bit time to be defined is the Prop\_Seg. Its length depends on the delay times measured in the system. A maximum bus length as well as a maximum node delay has to be defined for expandible CAN bus systems. The resulting time for Prop\_Seg is converted into time quanta (rounded up to the nearest integer multiple of  $t_q$ ).

The Sync\_Seg is 1  $t_q$  long (fixed), leaving (bit time – Prop\_Seg – 1)  $t_q$  for the two Phase Buffer Segments. If the number of remaining  $t_q$  is even, the Phase Buffer Segments have the same length, Phase\_Seg2 = Phase\_Seg1, else Phase\_Seg2 = Phase\_Seg1 + 1.

The minimum nominal length of Phase\_Seg2 has to be regarded as well. Phase\_Seg2 may not be shorter than any CAN controller's Information Processing Time in the network, which is device dependent and can be in the range of [0 ... 2]  $t_q$ .

The length of the Synchronization Jump Width is set to its maximum value, which is the minimum of 4 and Phase\_Seg1.

If more than one configurations are possible to reach a certain Bit rate, it is recommended to choose the configuration 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:

Tseg2 = Phase\_Seg2-1  
 Tseg1 = Phase\_Seg1+ Prop\_Seg-1  
 SJW = SynchronizationJumpWidth-1  
 BRP = Prescaler-1

### 22.4.2.2 Example for Bit Timing at High Baudrate

In this example, the frequency of CAN\_CLK is 10 MHz, BRP is 0, the bit rate is 1 MBit/s.

$t_q$	100	ns	=	$t_{CAN\_CLK}$	
delay of bus driver	60	ns			
delay of receiver circuit	40	ns			
delay of bus line (40m)	220	ns			
$t_{Prop}$	700	ns	=	$INT(2 \cdot \text{delays} + 1) = 7 \cdot t_q$	
$t_{SJW}$	100	ns	=	$1 \cdot t_q$	
$t_{TSeg1}$	800	ns	=	$t_{Prop} + t_{SJW}$	
$t_{TSeg2}$	100	ns	=	Information Processing Time + $1 \cdot t_q$	
$t_{Sync-Seg}$	100	ns	=	$1 \cdot t_q$	
bit time	1000	ns	=	$t_{Sync-Seg} + t_{TSeg1} + t_{TSeg2}$	
tolerance for CAN_CLK	1.58	%	=	$\frac{\min(TSeg1, TSeg2)}{2x(13x(\text{bit\_time} - TSeg2))}$	(33)
			=	$\frac{0.1 \mu s}{2x(13x(1 \mu s - 0.1 \mu s))}$	(34)

In this example, the concatenated bit time parameters are  $(1-1)_3 \& (8-1)_4 \& (1-1)_2 \& (1-1)_6$ , so the Bit Timing Register is programmed to= 0 x 00000700.

### 22.4.2.3 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.

$t_q$	100	ns	=	$2 \cdot t_{CAN\_CLK}$	
delay of bus driver	200	ns			
delay of receiver circuit	80	ns			
delay of bus line (40m)	220	ns			
$t_{Prop}$	1	$\mu s$	=	$1 \cdot t_q$	
$t_{SJW}$	4	$\mu s$	=	$4 \cdot t_q$	
$t_{TSeg1}$	5	$\mu s$	=	$t_{Prop} + t_{SJW}$	
$t_{TSeg2}$	3	$\mu s$	=	Information Processing Time + $3 \cdot t_q$	
$t_{Sync-Seg}$	1	$\mu s$	=	$1 \cdot t_q$	
bit time	9	$\mu s$	=	$t_{Sync-Seg} + t_{TSeg1} + t_{TSeg2}$	
tolerance for CAN_CLK	0.43	%	=	$\frac{\min(TSeg1, TSeg2)}{2x(13x(\text{bit\_time} - TSeg2))}$	(35)
			=	$\frac{3 \mu s}{2x(13x(9 \mu s - 3 \mu s))}$	(36)

In this example, the concatenated bit time parameters are  $(3-1)_3 \& (5-1)_4 \& (4-1)_2 \& (2-1)_6$ , so the Bit Timing Register is programmed to= 0x000024C1.



## 22.5 CAN Module Configuration

After a hardware reset all CAN protocol functions are disabled. The CAN module must be initialized and configured before it can participate on the CAN bus.

### 22.5.1 DCAN RAM Initialization through hardware

To start with a clean DCAN RAM, the complete DCAN RAM can be initialized with zeros and the ecc/parity bits set accordingly by configuring the following registers in the system module:

1. Memory Hardware Initialization Global Control Register (MINITGCR)
2. Memory Initialization Enable Register (MSINENA)

For more details on RAM hardware initialization support, refer to the system module reference guide.

### 22.5.2 CAN Module Initialization

To initialize the CAN Controller, the user has to set up the CAN Bit timing and those message objects which have to be used for CAN communication. Message objects that are not needed, can be deactivated.

So the two critical steps are:

1. Configuration of CAN Bit Timings
2. Configuration of Message Objects

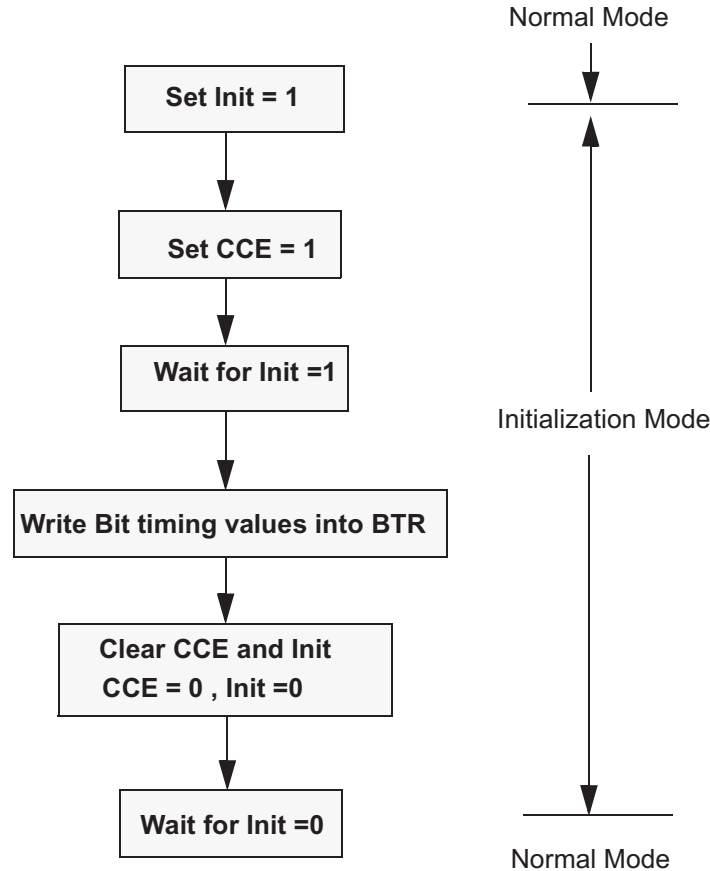
#### 22.5.2.1 Software Configuration of CAN Bit Timings

This step involves configuring the CAN baudrate register with the calculated CAN bit timing value.

The calculation procedure of CAN bit timing values for BTR register are mentioned in [Section 22.4](#).

Refer [Figure 22-4](#) for CAN bit timing software configuration flow.

**Figure 22-4. CAN Bit-timing Configuration**



**Step 1:** Enter “initialization mode” by setting the Init (Initialization) bit in the CAN Control Register.

While the Init bit is set, the message transfer from and to the CAN bus is stopped, and the status of the CAN\_TX output is recessive (high).

The CAN error counters are not updated. Setting the Init bit does not change any other configuration register.

Also note that the CAN module is also in initialization mode on hardware reset and during Bus-Off.

**Step 2:** Set the CCE (Configure Change Enable) bit in the CAN Control Register.

The access to the Bit Timing Register for the configuration of the Bit timing is enabled when both Init and CCE bits in the CAN Control Register are set.

**Step 3:** Wait for the Init bit to get set. This would make sure that the module has entered “Initialization mode”.

**Step 4:** Write the Bit-Timing values into the BTR (Bit-Timing Register) Register.

Refer to [Section 22.4.2.1](#) for BTR value calculation for a given bit-timing.

**Step 5:** Clear the CCE bit followed by Init bit.

**Step 6:** Wait for the Init bit to clear. This would make sure that the module has come out of “initialization mode”.

After step 6 (Init bit cleared), the module will attempt a synchronization on the CAN bus, provided that the BTR settings are meeting the CAN bus parameters.

---

**NOTE:** The module would not come out of the “initialization mode” if any incorrect BTR values are written in step 4.

---

### 22.5.2.2 Configuration of Message Objects

The whole Message RAM should be configured before putting the CAN into operation. All the message objects are deactivated by default. The user should configure the message object that are to be used to a particular identifier. The user can change the configuration of any message object or deactivate it when required.

The message objects can be configured only through the Interface registers (IFx) and the CPU does not have direct access to the message object (Message RAM) when DCAN is in operation.

To configure the message objects, the user must know about:

1. The message object structure ([Section 22.6](#))
2. The interface register set (IFx) ([Section 22.7](#))

---

**NOTE:** The message objects initialization is independent of the bit-timing configuration procedure.

---

## 22.6 Message RAM

The DCAN Message RAM contains message objects and parity bits for the message objects.

### 22.6.1 Structure of Message Objects

Figure 22-5 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 22-5. 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 22-2. Message Object Field Descriptions**

Name	Value	Description
MsgVal	0	Message valid The message object is ignored by the Message Handler.
	1	The message object is to be used by the Message Handler. Note: The CPU should reset the MsgVal bit of all unused Messages Objects during the initialization before it resets bit Init in the CAN Control Register. MsgVal must also be reset if the messages object is no longer used in operation. For reconfiguration of message objects during normal operation see <a href="#">Section 22.8.6</a> and <a href="#">Section 22.8.7</a> .
UMask	0	Use Acceptance Mask 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]	ID[28:0]	Message Identifier 29-bit ("extended") identifier bits
	ID[28:18]	11-bit ("standard") identifier bits
Msk[28:0]	0	Identifier Mask 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	0	Extended Identifier 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	0	Mask Extended Identifier 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	0	Message Direction 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.

**Table 22-2. Message Object Field Descriptions (continued)**

Name	Value	Description
	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	0	Mask Message Direction The message direction bit (Dir) has no effect on the acceptance filtering.
	1	The message direction bit (Dir) is used for acceptance filtering.
EOB	0	End of Block 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	0	New Data 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	0	Message Lost (only valid for message objects with direction = receive) 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	0	Receive Interrupt Enable IntPnd will not be triggered after the successful reception of a frame.
	1	IntPnd will be triggered after the successful reception of a frame.
TxIE	0	Transmit Interrupt Enable 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	0	Interrupt Pending 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	0	Remote Enable At the reception of a Remote Frame, TxRqst is not changed.
	1	At the reception of a Remote Frame, TxRqst is set.
TxRqst	0	Transmit Request 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]	0-8	Data Length Code 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.
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

**Table 22-2. Message Object Field Descriptions (continued)**

Name	Value	Description
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 <b>Note:</b> 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.

### 22.6.2 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. Writing to the address of an unimplemented message object may overwrite an implemented message object.

---



---

**NOTE:** The base address for DCAN1 RAM is 0xFF1E 0000, DCAN2 RAM is 0xFF1C 0000 and DCAN3 RAM is 0xFF1A 0000

---

Message Object number 1 has the highest priority.

**Table 22-3. Message RAM Addressing in Debug/Suspend and RDA mode**

Message Object number	offset from base address	word number	Debug/Suspend mode, see <a href="#">Section 22.6.3</a>	RDA mode, see <a href="#">Section 22.6.4</a>
1	0x0020	1	Parity	Data Bytes 4-7
	0x0024	2	MXtd,MDir,Mask	Data Bytes 0-3
	0x0028	3	Xtd,Dir,ID	ID[27:0],DLC
	0x002C	4	Ctrl	Mask,Xtd,Dir,ID[28]
	0x0030	5	Data Bytes 3-0	Parity,Ctrl,MXtd,MDir
	0x0034	6	Data Bytes 7-4	--
..	...	...	...	...
31	0x03E0	1	Parity	Data Bytes 4-7
	0x03E4	2	MXtd,MDir,Mask	Data Bytes 0-3
	0x03E8	3	Xtd,Dir,ID	ID[27]:0,DLC
	0x03EC	4	Ctrl	Mask,Xtd,Dir,ID[28]
	0x03F0	5	Data Bytes 3-0	Parity,Ctrl,MXtd,MDir
	0x03F4	6	Data Bytes 7-4	--
..	...	...	...	...
63	0x07E0	1	Parity	Data Bytes 4-7
	0x07E4	2	MXtd,MDir,Mask	Data Bytes 0-3
	0x07E8	3	Xtd,Dir,ID	ID[27:0],DLC
	0x07EC	4	Ctrl	Mask,Xtd,Dir,ID[28]
	0x07F0	5	Data Bytes 3-0	Parity,Ctrl,MXtd,MDir
	0x07F4	6	Data Bytes 7-4	--

**Table 22-3. Message RAM Addressing in Debug/Suspend and RDA mode (continued)**

64	0x0000	1	Parity	Data Bytes 4-7
	0x0004	2	MXtd,MDir,Mask	Data Bytes 0-3
	0x0008	3	Xtd,Dir,ID	ID[27]:0,DLC
	0x000C	4	Ctrl	Mask,Xtd,Dir,ID[28]
	0x0010	5	Data Bytes 3-0	Parity,Ctrl,MXtd,MDir
	0x0014	6	Data Bytes 7-4	--

### 22.6.3 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 22-6. Message RAM Representation in Debug/Suspend 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	reserved															
	reserved												Parity[4:0]			
MsgAddr + 0x04	MXtd	MDir	reser ved	Msk[28:16]												
	Msk[15:0]															
MsgAddr + 0x08	reser ved	Xtd	Dir	ID[28:16]												
	ID[15:0]															
MsgAddr + 0x0C	reserved															
	reser ved	MsgL st	reser ved	UMas k	TxIE	RxTE	RmtE n	reser ved	EOB	reserved				DLC[3:0]		
MsgAddr + 0x10	Data 3								Data 2							
	Data 1								Data 0							
MsgAddr + 0x14	Data 7								Data 6							
	Data 5								Data 4							

### 22.6.4 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. Before entering RDA mode it must be ensured that the Init bit is set to avoid any conflicts with the message handler accessing the message RAM.

**NOTE:** 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 22-7. 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	RxTE	RmtEn	EOB	MXtd	MDir	

---

**NOTE:** Write to unused bits has no effect.

---



## 22.7 Message Interface Register Sets

Accesses to the Message RAM are performed via the Interface Register sets.

1. Interface Register 1 and 2
2. Interface Register 3

The third Interface Register set IF3 can be configured to automatically receive control and user data from the Message RAM when a message object has been updated after reception of a CAN message. The CPU does not need to initiate the transfer from Message RAM to IF3 Register set.

The Message Handler avoids potential conflicts between concurrent accesses to Message RAM and CAN frame reception/transmission.

There are two modes where the Message RAM can be directly accessed by the CPU:

1. In Debug/Suspend mode (see [Section 22.6.3](#))
2. In RAM Direct Access (RDA) mode (see [Section 22.6.4](#))

For the Message RAM Base address, please refer to the device datasheet.

A complete message object (see [Section 22.6.1](#)) or parts of the message object may be transferred between the Message RAM and the IF1/IF2 Register set (see [Section 22.18.19](#)) in one single transfer.

### 22.7.1 Message Interface Register Sets 1 and 2

The Interface Register sets IF1 and IF2 provide indirect read/write access from the CPU to the Message RAM. The IF1 and IF2 register sets can buffer control and user data to be transferred to and from the message objects.

**Table 22-4. Message Interface Register Sets 1 and 2**

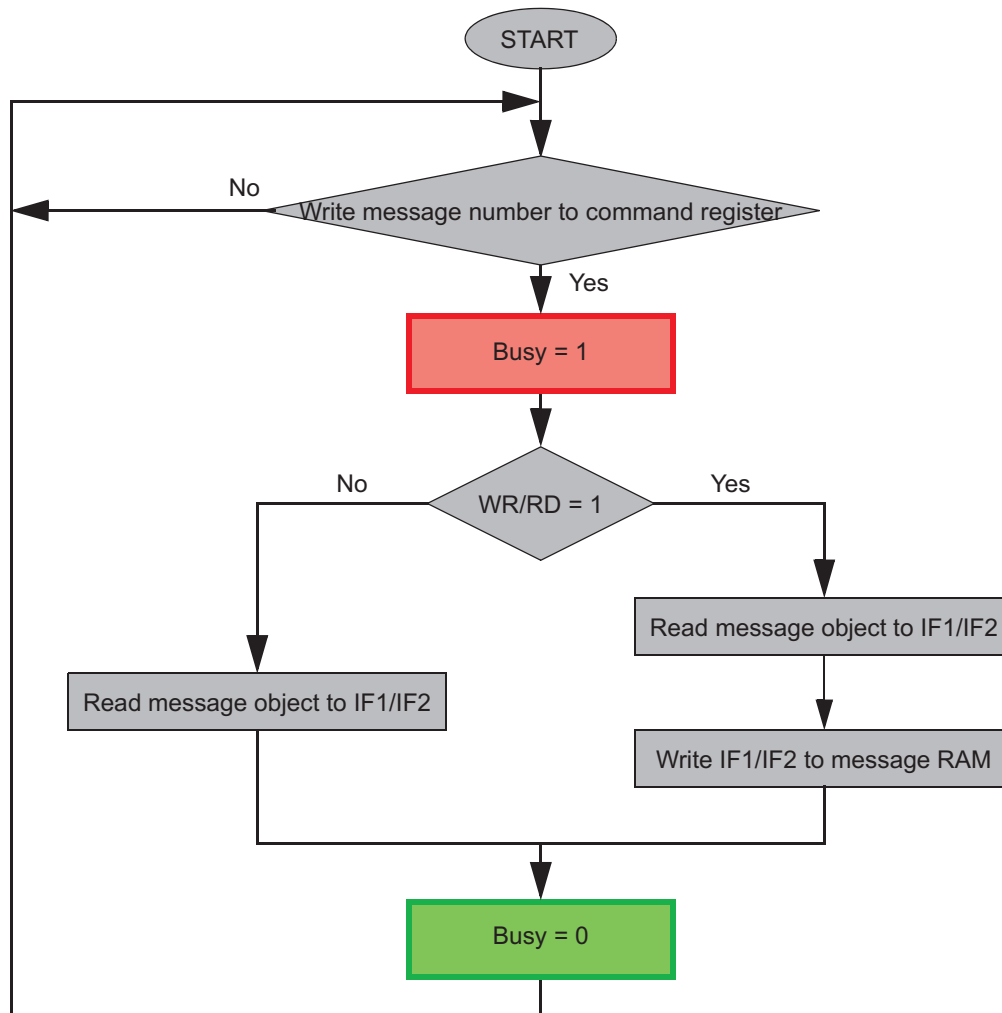
Address [CAN Base +]	IF1 Register Set		Address [CAN Base +]	IF2 Register Set	
	31	16 15 0		31	16 15 0
0x100	IF1 Command Mask	IF1 Command Request	0x120	IF2 Command Mask	IF2 Command Request
0x104	IF1 Mask 2	IF1 Mask 1	0x124	IF2 Mask 2	IF2 Mask 1
0x108	IF1 Arbitration 2	IF1 Arbitration 1	0x128	IF2 Arbitration 2	IF2 Arbitration 1
0x10C	res.	IF1 Message Control	0x12C	res.	IF2 Message Control
0x110	IF1 Data A 2	IF1 Data A 1	0x130	IF2 Data A 2	IF2 Data A 1
0x114	IF1 Data B 2	IF1 Data B 1	0x134	IF2 Data B 2	IF2 Data B 1

### 22.7.2 Using Message Interface Register Sets 1 and 2

The Command Register addresses the desired message object in the Message RAM and specifies whether a complete message object or only parts should be transferred. The data transfer is initiated by writing the message number to the bits [7:0] of the Command Register.

When the CPU initiates a data transfer between the IF1/IF2 Registers and Message RAM, the Message Handler sets the Busy bit in the respective Command Register to '1'. After the transfer has completed, the Busy bit is set back to '0' (see [Figure 22-8](#)).

Figure 22-8. Data Transfer Between IF1 / IF2 Registers and Message RAM



### 22.7.3 Message Interface Register 3

The IF3 register set can automatically be updated with received message objects without the need to initiate the transfer from Message RAM by CPU. The intention of this feature of IF3 is to provide an interface for the DMA to read packets efficiently.

Table 22-5. Message Interface Register 3

Address [CAN Base +]	IF3 Register Set	
	31	16 15 0
0x140	reserved	IF3 Observation
0x144	IF3 Mask 2	IF3 Mask 1
0x148	IF3 Arbitration 2	IF3 Arbitration 1
0x14C	reserved	IF3 Message Control
0x150	IF3 Data A 2	IF3 Data A 1
0x154	IF3 Data B 2	IF3 Data B 1
...		
0x160	IF3 Update Enable 2	IF3 Update Enable 1
0x164	IF3 Update Enable 4	IF3 Update Enable 3
0x168	IF3 Update Enable 6	IF3 Update Enable 5

**Table 22-5. Message Interface Register 3 (continued)**

Address	IF3 Register Set	
0x16C	IF3 Update Enable 8	IF3 Update Enable 7

The automatic update functionality can be programmed for each message object (see IF3 Update Enable Register, [Section 22.18.28](#)).

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, a DMA request is generated. The DMA request stays active until first read access to one of the IF3 registers. The DMA functionality has to be enabled by setting bit DE3 in CAN Control register. Please refer to the device datasheet to find out if this DMA source is available.

---

**NOTE:** The IF3 register set can not be used for transferring data into message objects.

---

## 22.8 Message Object Configurations

This section describes the possible message object configurations for CAN communication.

### 22.8.1 Configuration of a Transmit Object for Data Frames

Figure 22-9 shows how a Transmit Object can be initialized.

**Figure 22-9. 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 22.9.8. Identifier masking must be disabled (UMask = '0') if no Remote Frames are allowed to set the TxRqst bit (RmtEn = '0').

### 22.8.2 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.

### 22.8.3 Configuration of a Single Receive Object for Data Frames

Figure 22-10 shows how a Receive Object for Data Frames can be initialized.

**Figure 22-10. 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.

## 22.8.4 Configuration of a Single Receive Object for Remote Frames

Figure 22-11 shows how a Receive Object for Remote Frames can be initialized.

**Figure 22-11. 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 22.8.5](#).

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 22.9.8](#).

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.

## 22.8.5 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.

## 22.8.6 Reconfiguration of Message Objects for the Reception of Frames

A message object with Dir = '0' is configured for the reception of data frames, with Dir = '1' AND Umask = '1' AND RmtEn = '0' it is configured for the reception of remote frames.

It is necessary to reset MsgVal to not valid before changing any of the following configuration and control bits: ID[28:0], Xtd, Dir, DLC[3:0], RxIE, TxIE, RmtEn, EoB, Umask, Msk[28:0], MXtd, and MDir.

These parts of a message object may be changed without clearing MsgVal: Data[7:0], TxRqst, NewDat, MsgLst, and IntPnd.

## 22.8.7 Reconfiguration of Message Objects for the Reception of Frames

A message object with Dir = '1' AND (Umask = '0' OR RmtEn = '1') is configured for the transmission of data frames.

It is necessary to reset MsgVal to not valid before changing any of the following configuration and control bits: Dir, RxIE, TxIE, RmtEn, EoB, Umask, Msk[28:0], MXtd, and MDir

These parts of a message object may be changed without clearing MsgVal: ID[28:0], Xtd, DLC[3:0], Data[7:0], TxRqst, NewDat, MsgLst, and IntPnd.

## 22.9 Message Handling

When initialization is finished, the DCAN module synchronizes itself to the traffic on the CAN bus. It does acceptance filtering on received messages and stores those frames that are accepted into the designated message objects.

The application has to update the data of the messages to be transmitted and enable and request their transmission. The transmission is requested automatically when a matching Remote Frame is received.

The application may read messages 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 based on interrupts or by polling.

### 22.9.1 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.

### 22.9.2 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.

### 22.9.3 Transmission of Messages in Event Driven CAN Communication

If the shift register of the CAN Core is ready for loading and if there is no data transfer between the IFx Registers and Message RAM, the d bits in the Message Valid Register and the TxRqst bits in the Transmission Request Register are evaluated. The valid message object with the highest priority pending transmission request is loaded into the shift register by the Message Handler and the transmission is started. The message object's NewDat bit is reset.

After a successful transmission and if no new data was written to the message object (NewDat = '0') since the start of the transmission, the TxRqst bit will be reset. If TxIE is set, IntPnd will be set after a successful transmission. If the DCAN has lost the arbitration or if an error occurred during the transmission, the message will be retransmitted as soon as the CAN bus is free again. If meanwhile the transmission of a message with higher priority has been requested, the messages will be transmitted in the order of their priority.

If Automatic Retransmission mode is disabled by setting the DAR bit in the CAN Control Register, the behavior of bits TxRqst and NewDat in the Message Control Register of the Interface Register set is as follows:

- When a transmission starts, the TxRqst bit of the respective Interface Register set is reset, while bit NewDat remains set.
- When the transmission has been successfully completed, the NewDat bit is reset.

When a transmission failed (lost arbitration or error) bit NewDat remains set. To restart the transmission, the application has to set TxRqst again.

Received Remote Frames do not require a Receive Object. They will automatically trigger the transmission of a Data Frame, if in the matching Transmit Object the RmtEn bit is set.

### 22.9.4 Updating a Transmit Object

The CPU may update the data bytes of a Transmit Object any time via the IF1/IF2 Interface Registers, neither d nor TxRqst have to be reset before the update.

Even if only a part of the data bytes are to be updated, all four bytes in the corresponding IF1/IF2 Data A Register or IF1/IF2 Data B Register have to be valid before the content of that register is transferred to the message object. Either the CPU has to write all four bytes into the IF1/IF2 Data Register or the message object is transferred to the IF1/IF2 Data Register before the CPU writes the new data bytes.

When only the data bytes are updated, first 0x87 can be written to bits [23:16] of the Command Register and then the number of the message object is written to bits [7:0] of the Command Register, concurrently updating the data bytes and setting TxRqst with NewDat.

To prevent the reset of TxRqst at the end of a transmission that may already be in progress while the data is updated, NewDat has to be set together with TxRqst in event driven CAN communication. For details see [Section 22.9.3](#).

When NewDat is set together with TxRqst, NewDat will be reset as soon as the new transmission has started.

### 22.9.5 Changing a Transmit Object

If the number of implemented message objects is not sufficient to be used as permanent message objects only, the Transmit Objects may be managed dynamically. The CPU can write the whole message (Arbitration, Control, and Data) into the Interface Register. The bits [23:16] of the Command Register can be set to 0xB7 for the transfer of the whole message object content into the message object. Before changing the configuration of a message object, MsgVal has to be reset ( see [Section 22.8.7](#)).

If a previously requested transmission of this message object is not completed but already in progress, it will be continued; however it will not be repeated if it is disturbed.

To only update the data bytes of a message to be transmitted, bits [23:16] of the Command Register should be set to 0x87.



---

**NOTE:** After the update of the Transmit Object, the Interface Register set will contain a copy of the actual contents of the object, including the part that had not been updated.

---

### 22.9.6 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.

### 22.9.7 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.

### 22.9.8 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.

### 22.9.9 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.

### 22.9.10 Requesting New Data for a Receive Object

By means of a Remote Frame, the CPU may request another CAN node to provide new data for a receive object. Setting the TxRqst bit of a receive object will cause the transmission of a Remote Frame with the receive object's identifier. This Remote Frame triggers the other CAN node to start the transmission of the matching Data Frame. If the matching Data Frame is received before the Remote Frame could be transmitted, the TxRqst bit is automatically reset.

Setting the TxRqst bit without changing the contents of a message object requires the value 0x84 in bits [23:16] of the Command Register.

### 22.9.11 Storing Received Messages in FIFO Buffers

Several message objects may be grouped to form one or more FIFO Buffers. Each FIFO Buffer configured to store received messages with a particular (group of) Identifier(s). Arbitration and Mask Registers of the FIFO Buffer's message objects are identical. The EoB (End of Buffer) bits of all but the last of the FIFO Buffer's message objects are '0', in the last one the EoB bit is '1'.

Received messages with identifiers matching to a FIFO Buffer are stored into a message object of this FIFO Buffer, starting with the message object with the lowest message number.

When a message is stored into a message object of a FIFO Buffer the NewDat bit of this message object is set. By setting NewDat while EoB is '0' the message object is locked for further write accesses by the Message Handler until the CPU has cleared the NewDat bit.

Messages are stored into a FIFO Buffer until the last message object of this FIFO Buffer is reached. If none of the preceding message objects is released by writing NewDat to '0', all further messages for this FIFO Buffer will be written into the last message object of the FIFO Buffer (EoB = '1') and therefore overwrite previous messages in this message object.

### 22.9.12 Reading from a FIFO Buffer

Several messages may be accumulated in a set of message objects which are concatenated to form a FIFO Buffer before the application program is required (in order to avoid the loss of data) to empty the buffer.

A FIFO Buffer of length N will store N-1 plus the last received message since last time it was cleared.

A FIFO Buffer is cleared by reading and resetting the NewDat bits of all its message objects, starting at the FIFO Object with the lowest message number. This should be done in a subroutine following the example shown in [Figure 22-12](#).

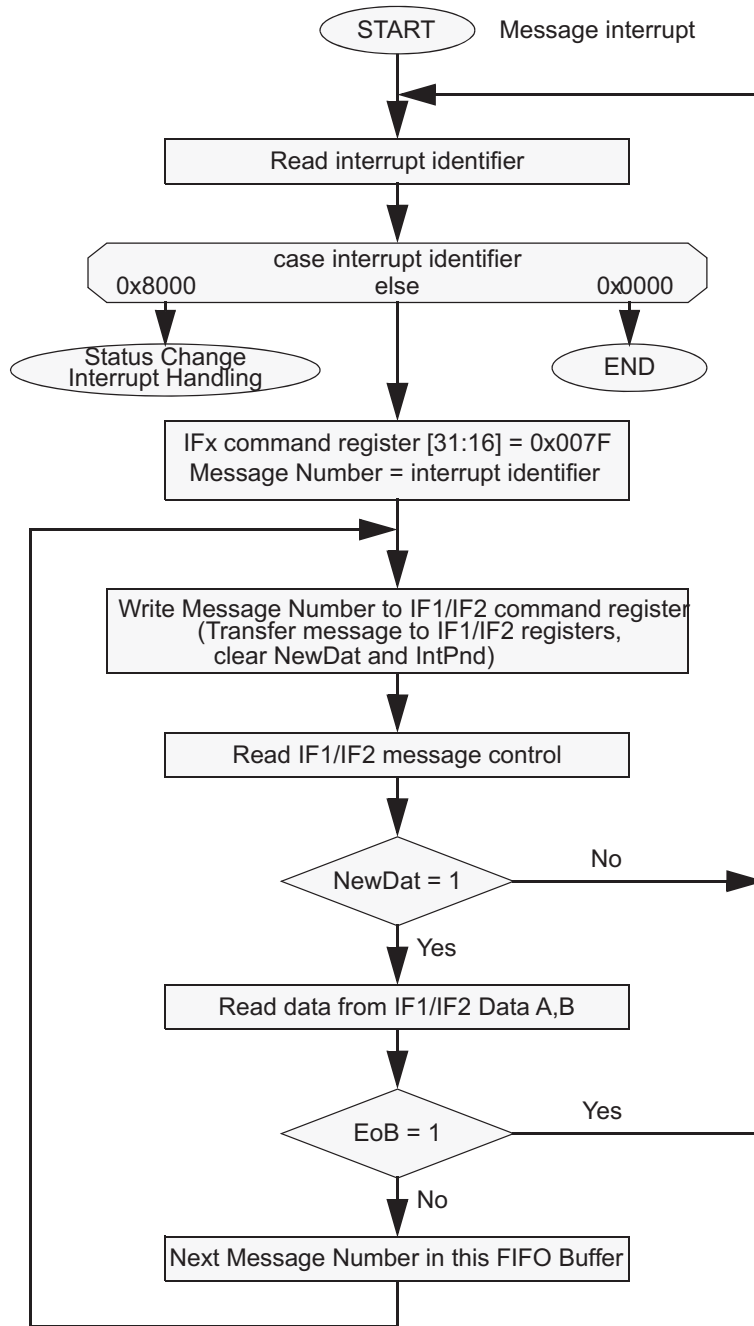
---

**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 22-12. CPU Handling of a FIFO Buffer (Interrupt Driven)



## 22.10 CAN Message Transfer

Once the DCAN is initialized and Initbit is reset to zero, the CAN Core synchronizes itself to the CAN bus and is ready for message transfer as per the configured message objects.

The CPU may enable the interrupt lines (setting IE0 and IE1 to '1') at the same time when it clears Init and CCE. The status interrupts EIE and SIE may be enabled simultaneously.

The CAN communication can be carried out in any of the following two modes:

1. Interrupt mode
2. Polling mode.

The Interrupt Register points to those message objects with IntPnd = '1'. It is updated even if the interrupt lines to the CPU are disabled (IE0 / IE1 are zero).

The CPU may poll all Message Object's NewDat and TxRqst bits in parallel from the NewData X Registers and the Transmission Request X Registers. Polling can be made easier if all Transmit Objects are grouped at the low numbers, all Receive Objects are grouped at the high numbers.

Received messages are stored into their appropriate message objects if they pass acceptance filtering.

The whole message (including all arbitration bits, DLC and up to eight data bytes) is stored into the message object. As a consequence, when 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 (For reconfiguration, see [Section 22.8.6](#))

If a permanent message object (arbitration and control bits set up during configuration and leaving unchanged for multiple CAN transfers) exists for the message, it is possible to only update the data bytes.

If several transmit messages should be assigned to one message object, the whole message object has to be configured before the transmission of this message is requested.

The transmission of multiple message objects may be requested at the same time. They are subsequently transmitted, according to their internal priority.

Messages may be updated or set to not valid at any time, even if a requested transmission is still pending (for reconfiguration see [Section 22.8.7](#)). However, the data bytes will be discarded if a message is updated before a pending transmission has started.

Depending on the configuration of the message object, a transmission may be automatically requested by the reception of a remote frame with a matching identifier.

### 22.10.1 Automatic Retransmission

According to the CAN Specification (ISO11898), 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(Disable Automatic Retransmission) in CAN Control Register. Further details to this mode are provided in [Section 22.9.3](#).

### 22.10.2 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. 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.

**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.

---

## 22.11 Interrupt Functionality

Interrupts can be generated on two interrupt lines:

1. DCAN0INT line
2. DCAN1INT line

These lines can be enabled by setting IE0 and IE1 bits respectively in CAN Control Register.

The DCAN provides three groups of interrupt sources: Message Object Interrupts, Status Change Interrupts and Error Interrupts. (see [Figure 22-13](#) and [Figure 22-14](#))

The source of an interrupt can be determined by the interrupt identifiers Int0ID / Int1ID in the Interrupt Register (see [Section 22.18.5](#)). When no interrupt is pending, the register will hold the value zero.

Each interrupt line remains active until the dedicated field in the Interrupt Register DCAN INT (Int0ID / Int1ID) again reach zero (this means the cause of the interrupt is reset), or until IE0 / IE1 are reset.

The value 0x8000 in the Int0ID field indicates that an interrupt is pending because the CAN Core has updated (not necessarily changed) the Error and Status Register (Error Interrupt or Status Interrupt). This interrupt has the highest priority. The CPU can update (reset) the status bits WakeUpPnd, RxOk, TxOk and LEC by reading the Error and Status Register DCAN ES, but a write access of the CPU will never generate or reset an interrupt.

Values between 1 and the number of the last message object indicates that the source of the interrupt is one of the message objects, Int0ID resp. Int1ID will point to the pending message interrupt with the highest priority. The Message Object 1 has the highest priority, the last message object has the lowest priority.

An interrupt service routine 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.

### 22.11.1 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 22.6.1](#).

Message Object interrupts can be routed to either DCAN0INT or DCAN1INT line, controlled by the Interrupt Multiplexer Register, see [Section 22.18.17](#).

### 22.11.2 Status Change Interrupts

The events WakeUpPnd, RxOk, TxOk and LEC in Error and Status Register (DCAN ES) belong to the Status Change Interrupts. The Status Change Interrupt group can be enabled by bit in CAN Control Register.

If SIE is set, a Status Change Interrupt will be generated at each CAN frame, independent of bus errors or valid CAN communication, and also independent of the Message RAM configuration.

Status Change interrupts can only be routed to interrupt line DCAN0INT 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. (Additional information can be found in [Section 22.12.2](#)).

---

### 22.11.3 Error Interrupts

The events PER, BOff and EWarn (monitored in Error and Status Register, DCAN ES) belong to the Error Interrupts. The Error Interrupt group can be enabled by setting bit EIE in CAN Control Register.

Error interrupts can only be routed to interrupt line DCAN0INT which has to be enabled by setting IE0 in the CAN Control Register.

Figure 22-13. CAN Interrupt Topology 1

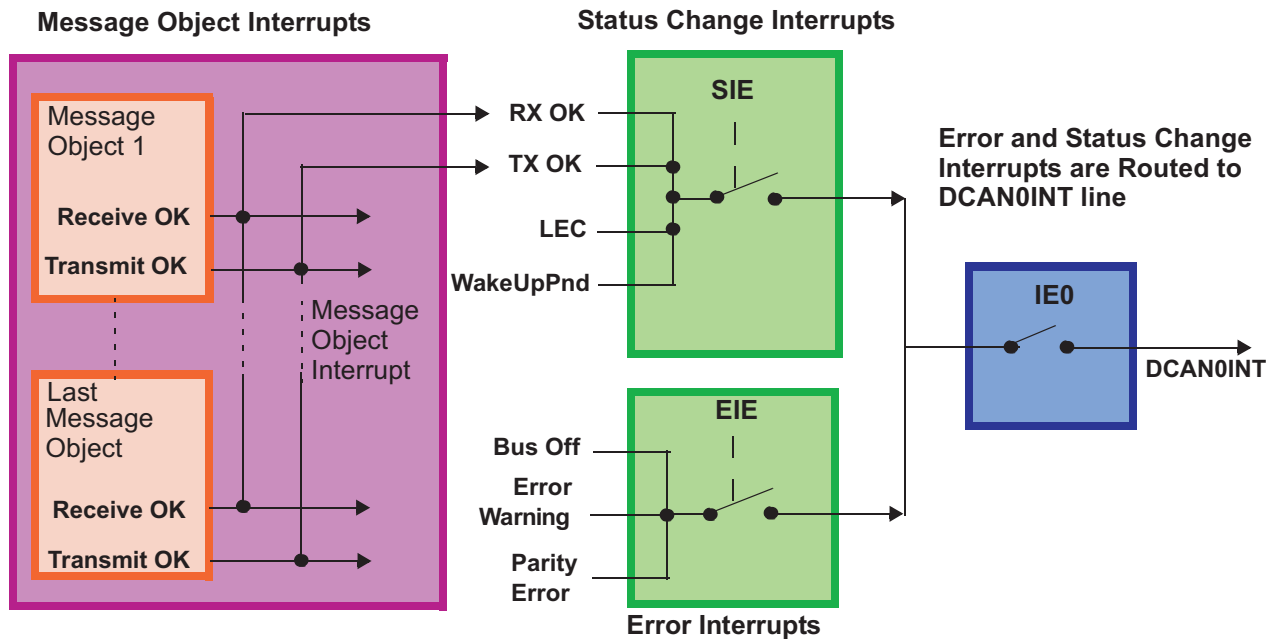
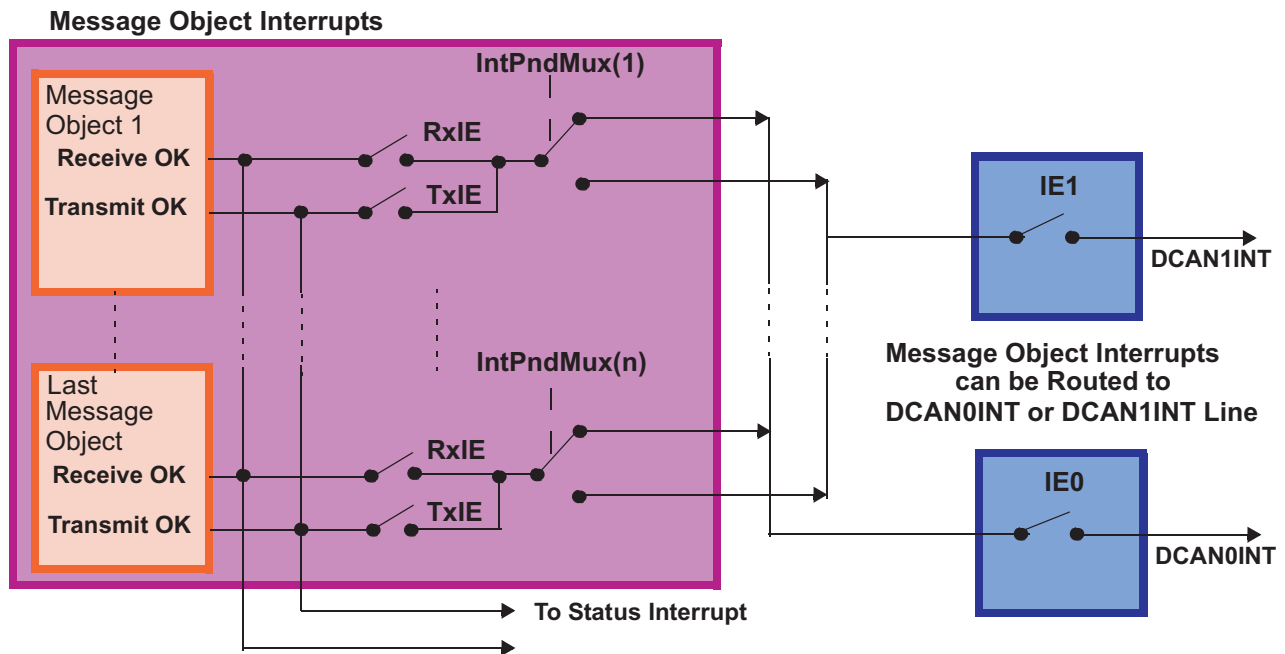


Figure 22-14. CAN Interrupt Topology 2



## 22.12 Global Power Down Mode

The device architecture supports a centralized global power down control over the peripheral modules through the Peripheral Central Resource (PCR) module (Additional information can be found in Platform Architecture Specification).

### 22.12.1 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 Initbit to indicate that the global power down mode has been entered.

### 22.12.2 Wakeup From Global Power Down Mode

When the DCAN module is in global power down mode, a CAN bus activity detection circuit exists, which can be active, if enabled. If this circuit is active, on occurrence of a dominant CAN bus level, the DCAN will set the **WakeUpPnd** bit in Error and Status Register (DCAN ES).

If Status Interrupts are enabled, also an interrupt will be generated. This interrupt could be used by the application to wakeup the DCAN. For this, the application needs to set the appropriate Peripheral Power Down Clear bit (**PSPWRDWNCLR<sub>x</sub>**) 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.

---

## 22.13 Local Power Down Mode

Besides from the centralized power down mechanism controlled by the PCR module (global power down, see [Section 22.16](#)), the DCAN supports a local power down mode which can be controlled within the DCAN control registers.

### 22.13.1 Entering Local Power Down Mode

The local power down mode is requested by setting the PDRbit 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 Initbit 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 22.13.2](#)) 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.

---

### 22.13.2 Wakeup From Local Power Down

There are two ways to wake up the DCAN from local power down mode:

1. The application could wake up the DCAN module manually by clearing the PDR bit and then clearing the Init bit in CAN Control Register.
2. Alternatively, a CAN bus activity detection circuit can be activated by setting the wake up on bus activity bit (WUBA) in CAN Control Register. If this circuit is active, on occurrence of a dominant CAN bus level, the DCAN will automatically start the wake up sequence. It will clear the PDR bit in CAN Control Register and also clear the PDA bit in Error and Status Register. The WakeUpPnd bit in CAN Error and Status Register will be set. If Status Interrupts are enabled, also an interrupt will be generated. Finally the Init bit in CAN control register will be cleared.

After the Init bit has been cleared, the module waits until it detects 11 consecutive recessive bits on the CAN\_RX pin and then goes Bus-Active again.

---

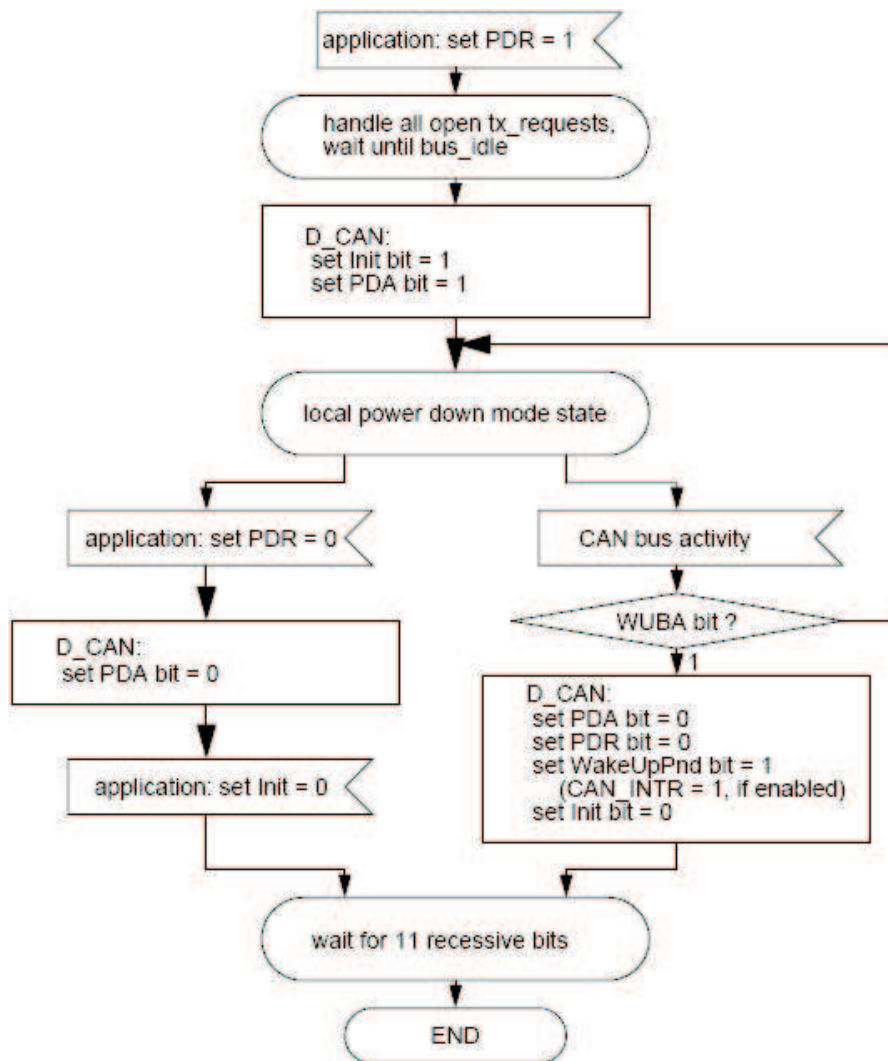
**NOTE:** The CAN transceiver circuit has to stay active while CAN bus observation. The first CAN message, which initiates the bus activity, cannot be received. This means that the first message received in power down and automatic wake-up mode, is lost.

---

[Figure 22-15](#) shows a flow diagram about entering and leaving local power down mode.



Figure 22-15. Local Power Down Mode Flow Diagram



## 22.14 GIO support

The CAN\_RX and CAN\_TX pins of each DCAN module can be used as general purpose IO pins, if CAN functionality is not needed. This function is controlled by the CAN TX IO Control register (see [Section 22.18.29](#)) and the CAN RX IO Control register (see [Section 22.18.30](#)).

## 22.15 Test Modes

The DCAN provides several test modes which are mainly intended for production tests or self test.

For all test modes, Test bit in the CAN Control Register needs to be set to one. This enables write access to the Test Register.

---

**NOTE:** When using any of the Loop Back modes, it must be ensured by software that all message transfers are finished before setting the Init bit to '1'.

---

### 22.15.1 Silent Mode

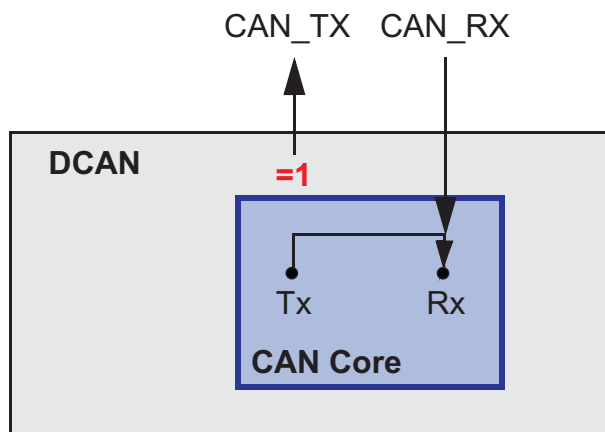
The Silent Mode may be used to analyze the traffic on the CAN bus without affecting it by sending dominant bits (e.g. acknowledge bit, overload flag, active error flag). The DCAN is still able to receive valid data frames and valid remote frames, but it will not send any dominant bits. However, these are internally routed to the CAN Core.

Figure 22-16 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 Silentbit in Test Register to one.

In ISO 11898-1, the Silent Mode is called the Bus Monitoring Mode.

**Figure 22-16. CAN Core in Silent Mode**



### 22.15.2 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 22-17 shows the connection of signals CAN\_TX and CAN\_RX to the CAN Core in Loop Back Mode.

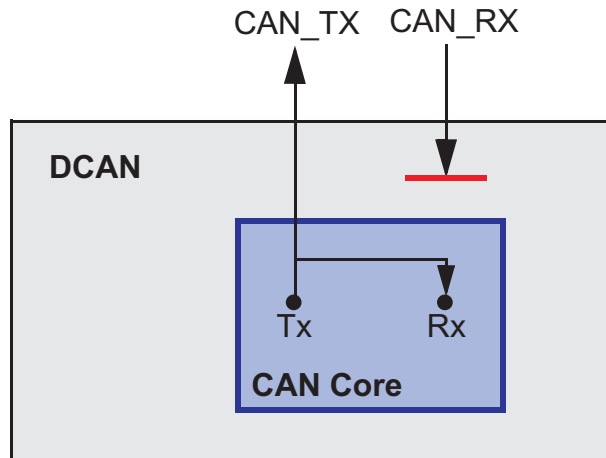
Loop Back Mode can be activated by setting bit LBackin Test Register to one.

---

**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 22.15.3).

---

Figure 22-17. CAN Core in Loop Back Mode



### 22.15.3 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.

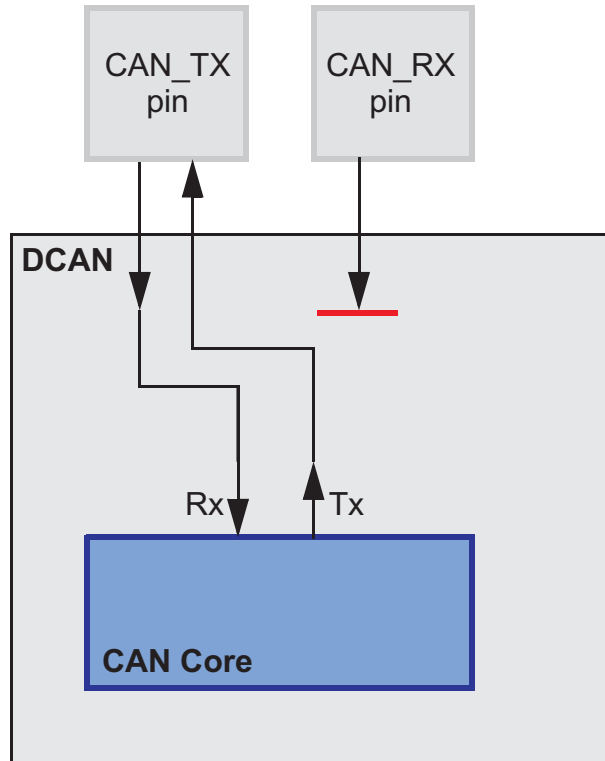
External Loop Back Mode can be activated by setting bit **ExL** in Test Register to one.

Figure 22-18 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.

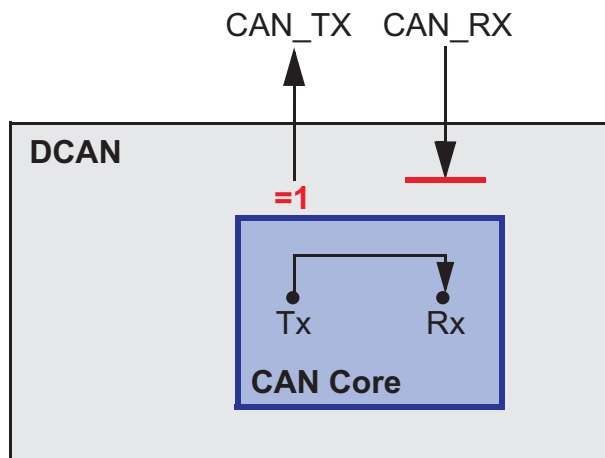
---

**Figure 22-18. CAN Core in External Loop Back Mode**


#### 22.15.4 Loop Back Combined with Silent Mode

It is also possible to combine Loop Back Mode and Silent Mode by setting bits LBackand 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 22-19 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 22-19. CAN Core in Loop Back Combined with Silent Mode**


### 22.15.5 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 22.18.6](#).

---

**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.

---

## 22.16 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 22.6.4](#).

Parity information is stored in the Message RAM on write accesses and will be checked against the stored parity bit from Message RAM on read accesses.

The Parity check functionality can be enabled or disabled by PMD bit field in CAN Control Register.

In case of disabled parity check, the parity bits in message RAM will be left unchanged on write access to data area and no check will be done on read access.

If parity checking is enabled, parity bits will be automatically generated and checked by the DCAN. The parity bits could be read in Debug/Suspend mode (see [Section 22.6.3](#)) or in RDA mode (see [Section 22.6.4](#)). However, direct write access to the parity bits is only possible in this two modes with parity check disabled.

A parity bit will be set, if the modulo-2-sum of the data bits is 1. This definition is equivalent to: The parity bit will be set, if the number of 1 bits in the data is odd.

---

**NOTE:** Parity scheme can be changed via the System module DEV Parity Control Register1 on device basis for all peri RAMs.

---

### 22.16.1 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 d bit of the message object will be reset.

The message object data can be read by the host CPU, independently of parity errors. Thus, the application has to ensure that the read data is valid, e.g. by immediately checking the Parity Error Code register on parity error interrupt.

---

**NOTE:** During RAM initialization, no parity check will be done.

---

### 22.16.2 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.

---

## 22.17 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 IDSin CAN Control Register.

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 22.6.3](#)).

---

**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)



## 22.18 DCAN Control Registers

**Table 22-6. DCAN Control Registers**

Offset	Acronym	Register Description	Section
00	DCAN CTL	CAN Control Register	<a href="#">Section 22.18.1</a>
04h	DCAN ES	Error and Status Register	<a href="#">Section 22.18.2</a>
08h	DCAN ERRC	Error Counter Register	<a href="#">Section 22.18.3</a>
0Ch	DCAN BTR	Bit Timing Register	<a href="#">Section 22.18.4</a>
10h	DCAN INT	Interrupt Register	<a href="#">Section 22.18.5</a>
14h	DCAN TEST	Test Register	<a href="#">Section 22.18.6</a>
1Ch	DCAN PERR	Parity Error Code Register	<a href="#">Section 22.18.7</a>
80h	DCAN ABOTR	Auto-Bus-On Time Register	<a href="#">Section 22.18.8</a>
84h	DCAN TXRQX	Transmission Request X Register	<a href="#">Section 22.18.9</a>
88h	DCAN TXRQ12	Transmission Request 12 Register	<a href="#">Section 22.18.10</a>
8Ch	DCAN TXRQ34	Transmission Request 34 Register	<a href="#">Section 22.18.10</a>
90h	DCAN TXRQ56	Transmission Request 56 Register	<a href="#">Section 22.18.10</a>
94h	DCAN TXRQ78	Transmission Request 78 Register	<a href="#">Section 22.18.10</a>
98h	DCAN NWDATX	New Data X Register	<a href="#">Section 22.18.11</a>
9Ch	DCAN NWDAT12	New Data 12 Register	<a href="#">Section 22.18.12</a>
A0h	DCAN NWDAT34	New Data 34 Register	<a href="#">Section 22.18.12</a>
A4h	DCAN NWDAT56	New Data 56 Register	<a href="#">Section 22.18.12</a>
A8h	DCAN NWDAT78	New Data 78 Register	<a href="#">Section 22.18.12</a>
ACh	DCAN INTPNDX	Interrupt Pending X Register	<a href="#">Section 22.18.13</a>
B0h	DCAN INTPND12	Interrupt Pending 12 Register	<a href="#">Section 22.18.14</a>
B4h	DCAN INTPND34	Interrupt Pending 34 Register	<a href="#">Section 22.18.14</a>
B8h	DCAN INTPND56	Interrupt Pending 56 Register	<a href="#">Section 22.18.14</a>
BCh	DCAN INTPND78	Interrupt Pending 78 Register	<a href="#">Section 22.18.14</a>
C0h	DCAN MSGVALX	Message Valid X Register	<a href="#">Section 22.18.15</a>
C4h	DCAN MSGVAL12	Message Valid 12 Register	<a href="#">Section 22.18.16</a>
C8h	DCAN MSGVAL34	Message Valid 34 Register	<a href="#">Section 22.18.16</a>
CCh	DCAN MSGVAL56	Message Valid 56 Register	<a href="#">Section 22.18.16</a>
D0h	DCAN MSGVAL78	Message Valid 78 Register	<a href="#">Section 22.18.16</a>
D8h	DCAN INTMUX12	Interrupt Multiplexer 12 Register	<a href="#">Section 22.18.17</a>
DCh	DCAN INTMUX34	Interrupt Multiplexer 34 Register	<a href="#">Section 22.18.17</a>
E0h	DCAN INTMUX56	Interrupt Multiplexer 56 Register	<a href="#">Section 22.18.17</a>
E4h	DCAN INTMUX78	Interrupt Multiplexer 78 Register	<a href="#">Section 22.18.17</a>
100h	DCAN IF1CMD	IF1 Command Register	<a href="#">Section 22.18.18</a>
104h	DCAN IF1MSK	IF1 Mask Register	<a href="#">Section 22.18.19</a>
108h	DCAN IF1ARB	IF1 Arbitration Register	<a href="#">Section 22.18.20</a>
10Ch	DCAN IF1MCTL	IF1 Message Control Register	<a href="#">Section 22.18.21</a>
110h	DCAN IF1DATA	IF1 Data A Register	<a href="#">Section 22.18.22</a>
114h	DCAN IF1DATB	IF1 Data B Register	<a href="#">Section 22.18.22</a>
120h	DCAN IF2CMD	IF2 Command Register	<a href="#">Section 22.18.18</a>
124h	DCAN IF2MSK	IF2 Mask Register	<a href="#">Section 22.18.19</a>
128h	DCAN IF2ARB	IF2 Arbitration Register	<a href="#">Section 22.18.20</a>
12Ch	DCAN IF2MCTL	IF2 Message Control Register	<a href="#">Section 22.18.21</a>
130h	DCAN IF2DATA	IF2 Data A Register	<a href="#">Section 22.18.22</a>
134h	DCAN IF2DATB	IF2 Data B Register	<a href="#">Section 22.18.22</a>
140h	DCAN IF3OBS	IF3 Observation Register	<a href="#">Section 22.18.23</a>
144h	DCAN IF3MSK	IF3 Mask Register	<a href="#">Section 22.18.24</a>

**Table 22-6. DCAN Control Registers (continued)**

<b>Offset</b>	<b>Acronym</b>	<b>Register Description</b>	<b>Section</b>
148h	DCAN IF3ARB	IF3 Arbitration Register	<a href="#">Section 22.18.25</a>
14Ch	DCAN IF3MCTL	IF3 Message Control Register	<a href="#">Section 22.18.26</a>
150h	DCAN IF3DATA	IF3 Data A Register	<a href="#">Section 22.18.27</a>
154h	DCAN IF3DATB	IF3 Data B Register	<a href="#">Section 22.18.27</a>
160h	DCAN IF3UPD12	IF3 Update Enable 12 Register	<a href="#">Section 22.18.28</a>
164h	DCAN IF3UPD34	IF3 Update Enable 34 Register	<a href="#">Section 22.18.28</a>
168h	DCAN IF3UPD56	IF3 Update Enable 56 Register	<a href="#">Section 22.18.28</a>
16Ch	DCAN IF3UPD78	IF3 Update Enable 78 Register	<a href="#">Section 22.18.28</a>
1E0h	DCAN TIOC	CAN TX IO Control Register	<a href="#">Section 22.18.29</a>
1E4h	DCAN RIOC	CAN RX IO Control Register	<a href="#">Section 22.18.30</a>

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'.

## 22.18.1 CAN Control Register (DCAN CTL)

**Figure 22-20. CAN Control Register (DCAN CTL) [offset = 0]**

31		Reserved				26	25	24
		R-0					WUBA	PDR
							R/W-0	R/W-0
23		21	20	19	18	17	16	
Reserved		DE3	DE2	DE1	IE1	InitDbg		
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R-0		
15		14	13	10		9	8	
SWR	Reserved	PMD			ABO	IDS		
R/W-0	R-0	R/W-5h			R/W-0	R/W-0		
7		6	5	4	3	2	1	0
Test	CCE	DAR	Reserved	EIE	SIE	IE0	Init	
R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-1	

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

**Table 22-7. CAN Control Register Field Descriptions**

Bit	Field	Value	Description
31-26	Reserved	0	These bits are always read as 0. Writes have no effect.
25	WUBA	0	Automatic wake up on bus activity when in local power down mode No detection of a dominant CAN bus level while in local power down mode.
		1	Detection of a dominant CAN bus level while in local power down mode is enabled. On occurrence of a dominant CAN bus level, the wake up sequence is started. (Additional information can be found in <a href="#">Section 22.13</a> .) <b>Note:</b> 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	0	Request for local low power down mode No application request for local low power down mode. If the application has cleared this bit while DCAN in local power down mode, also the Init bit has to be cleared.
		1	Local power down mode has been requested by application. The DCAN will acknowledge the local power down mode by setting bit PDA in Error and Status Register. The local clocks will be turned off by DCAN internal logic (Additional information can be found in <a href="#">Section 22.13</a> ).
23-21	Reserved	0	These bits are always read as 0. Writes have no effect.
20	DE3	0	Enable DMA request line for IF3 Disabled
		1	Enabled <b>Note:</b> A pending DMA request for IF3 remains active until first access to one of the IF3 registers.
19	DE2	0	Enable DMA request line for IF2 Disabled
		1	Enabled <b>Note:</b> A pending DMA request for IF2 remains active until first access to one of the IF2 registers.
18	DE1	0	Enable DMA request line for IF1 Disabled
		1	Enabled <b>Note:</b> A pending DMA request for IF1 remains active until first access to one of the IF1 registers.
17	IE1	0	Interrupt line 1 Enable Disabled - Module Interrupt DCAN1INT is always low.
		1	Enabled - Interrupts will assert line DCAN1INT to one; line remains active until pending interrupts are processed.

**Table 22-7. CAN Control Register Field Descriptions (continued)**

Bit	Field	Value	Description
16	InitDbg	0	Internal Init state while debug access 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	0	SW Reset Enable 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. <b>Note:</b> 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	5h	Parity on/off Parity function disabled
		Others	Parity function enabled
9	ABO	0	Auto-Bus-On Enable The Auto-Bus-On feature is disabled
		1	The Auto-Bus-On feature is enabled
8	IDS	0	Interruption Debug Support Enable 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	0	Test Mode Enable Normal Operation
		1	Test Mode
6	CCE	0	Configuration Change Enable 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	0	Disable Automatic Retransmission Automatic Retransmission of not successful messages enabled.
		1	Automatic Retransmission disabled.
4	Reserved	0	This bit is always read as 0. Writes have no effect.
3	EIE	0	Error Interrupt Enable Disabled - PER, BOff and EWarn bits cannot generate an interrupt.
		1	Enabled - PER, BOff and EWarn bits can generate an interrupt at DCAN0INT line and affect the Interrupt Register.
2	SIE	0	Status Change Interrupt Enable Disabled - WakeUpPnd, RxOk, TxOk and LEC bits cannot generate an interrupt.
		1	Enabled - WakeUpPnd, RxOk, TxOk and LEC can generate an interrupt at DCAN0INT line and affect the Interrupt Register.
1	IE0	0	Interrupt line 0 Enable 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	0	Initialization Normal Operation
		1	Initialization mode is entered

**NOTE:** The Bus-Off recovery sequence (see CAN specification) cannot be shortened by setting or resetting Init bit. If the module goes Bus-Off, it will automatically set the Init bit and stop all bus activities.

When the Init bit is cleared by the application again, the module will then wait for 129 occurrences of Bus Idle (129 \* 11 consecutive recessive bits) before resuming normal operation. At the end of the Bus-Off recovery sequence, the error counters will be reset.

After the Init bit is reset, each time when a sequence of 11 recessive bits is monitored, a Bit0 error code is written to the Error and Status Register, enabling the CPU to check whether the CAN bus is stuck at dominant or continuously disturbed, and to monitor the proceeding of the Bus-Off recovery sequence.

---

**22.18.2 Error and Status Register (DCAN ES)**
**Figure 22-21. Error and Status Register (DCAN ES) [offset = 4h]**

	Reserved	
	R-0	
15	Reserved	8
	R-0	R-0
7	EWarn	PER
	R-0	R-7h
6	EPass	RxOK
	R-0	R-0
5	RxBOK	TxOK
	R-0	R-0
4	PDA	WakeUpPnd
	R-0	R-0
3	Boff	LEC
	R-0	R-0
2	Reserved	Reserved
	R-0	R-0
1	Reserved	Reserved
	R-0	R-0
0	Reserved	Reserved
	R-0	R-0

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

**Table 22-8. Error and Status Register Field Descriptions**

Bit	Field	Value	Description
31-11	Reserved	0	These bits are always read as 0. Writes have no effect.
10	PDA	0	Local power down mode acknowledge 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	WakeUp Pnd	0	Wake Up Pending This bit can be used by the CPU to identify the DCAN as the source to wake up the system. 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	0	Parity Error Detected 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	0	Bus-Off State The CAN module is not Bus-Off state.
		1	The CAN module is in Bus-Off state.
6	EWarn	0	Warning State 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	0	Error Passive State 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	0	Received a message successfully 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	0	Transmitted a message successfully 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 22-8. Error and Status Register Field Descriptions (continued)**

Bit	Field	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 (8000h) in the Interrupt Register will be replaced by the next lower priority interrupt value.

---

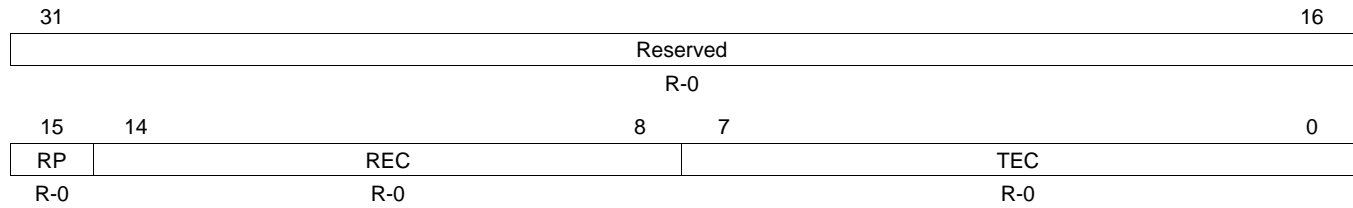


---

**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.

---

### 22.18.3 Error Counter Register (DCAN ERRC)

**Figure 22-22. Error Counter Register (DCAN ERRC) [offset = 8h]**


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

**Table 22-9. Error Counter Register Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	These bits are always read as 0. Writes have no effect.
15	RP	0	Receive Error Passive 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		Receive Error Counter. Actual state of the Receive Error Counter. (values from 0 to 255).
7-0	TEC		Transmit Error Counter. Actual state of the Transmit Error Counter. (values from 0 to 255).



## 22.18.4 Bit Timing Register (DCAN BTR)

**Figure 22-23. Bit Timing Register (DCAN BTR) [offset = Ch]**

31	Reserved												20	19	BRPE		16
R-0												R/W-0					
15	14	12		11	8			7	6	5	BRP		0				
Rsvd.	TSeg2		TSeg1			SJW		BRP									
R-0	R/W-2h		R/W-3h			R/W-0		R/W-1h									

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

**Table 22-10. Bit Timing Register Field Descriptions**

Bit	Field	Value	Description
31-20	Reserved	0	These bits are always read as 0. Writes have no effect.
	BRPE	0-Fh	Baud Rate Prescaler Extension. Valid programmed values are 0 to 15. By programming BRPE the Baud Rate Prescaler can be extended to values up to 1024.
15	Reserved	0	This bit is always read as 0. Writes have no effect.
14-12	TSeg2	0-7h	Time segment after the sample point Valid programmed values are 0 to 7. The actual TSeg2 value which is interpreted for the Bit Timing will be the programmed TSeg2 value + 1.
11-8	TSeg1	1h-Fh	Time segment before the sample point Valid programmed values are 1 to 15. The actual TSeg1 value interpreted for the Bit Timing will be the programmed TSeg1 value + 1.
7-6	SJW	0-3h	Synchronization Jump Width Valid programmed values are 0 to 3. The actual SJW value interpreted for the Synchronization will be the programmed SJW value + 1.
5-0	BRP	0-3Fh	Baud Rate Prescaler Value by which the CAN_CLK frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid programmed values are 0 to 63. The actual BRP value interpreted for the Bit Timing will be the programmed BRP value + 1.

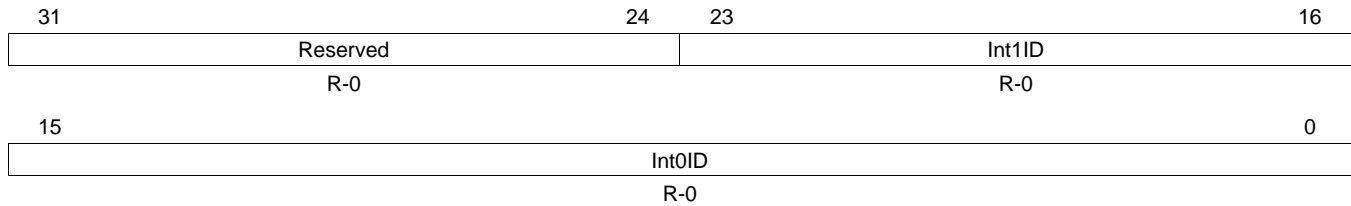
**NOTE:** This register is only writable if CCE and Init bits in the CAN Control Register are set.

**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 to 1024 CAN\_CLK periods.

With a CAN\_CLK of 8 MHz and BRPE = 00, the reset value of 2301h configures the DCAN for a bit rate of 500kBit/s.

For details see [Section 22.4.2.1](#).

**22.18.5 Interrupt Register (DCAN INT)**
**Figure 22-24. Interrupt Register (DCAN INT) [offset = 10h]**


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

**Table 22-11. Interrupt Register Field Descriptions**

Bit	Field	Value	Description
31-24	Reserved	0	These bits are always read as 0. Writes have no effect.
23-16	Int1ID	0 1h-80h 81h-FFh	<p>Interrupt 1 Identifier (indicates the message object with the highest pending interrupt)</p> <p>No interrupt is pending</p> <p>Number of message object which caused the interrupt.</p> <p>Unused</p> <p>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.</p> <p>A message interrupt is cleared by clearing the message object's IntPnd bit.</p> <p>Among the message interrupts, the message object's interrupt priority decreases with increasing message number.</p>
15-0	Int0ID	0 1h-80h 81h-7FFFh 8000h 8001h-FFFFh	<p>Interrupt Identifier (the number here indicates the source of the interrupt)</p> <p>No interrupt is pending</p> <p>Number of message object which caused the interrupt.</p> <p>Unused</p> <p>Error and Status Register value is not 0x07.</p> <p>Unused</p> <p>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.</p> <p>The Status Interrupt has the highest priority. Among the message interrupts, the message object's interrupt priority decreases with increasing message number.</p>

## 22.18.6 Test Register (DCAN TEST)

**Figure 22-25. Test Register (DCAN TEST) [offset = 14h]**

31	Reserved										20	19	16
R-0													
15	Reserved			RDA	EXL	Rx	Tx		LBack	Silent	Reserved		0
R-0			R/W-0	R/W-0	R-U	R/W-0		R/W-0	R/W-0	R-0			

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

**Table 22-12. Test Register Field Descriptions**

Bit	Field	Value	Description
31-10	Reserved	0	These bits are always read as 0. Writes have no effect.
9	RDA	0	Normal Operation
		1	Direct access to the RAM is enabled while in Test Mode.
8	EXL	0	Disabled
		1	Enabled
7	Rx	0	The CAN bus is dominant.
		1	The CAN bus is recessive.
6-5	Tx	0	Normal operation, CAN_TX is controlled by the CAN Core.
		1h	Sample Point can be monitored at CAN_TX pin.
		2h	CAN_TX pin drives a dominant value.
		3h	CAN_TX pin drives a recessive value.
4	LBack	0	Disabled
		1	Enabled
3	Silent	0	Disabled
		1	Enabled
2-0	Reserved	0	These bits are always read as 0. Writes have no effect.

For all test modes, the Test bit in CAN Control Register needs to be set to 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.

---

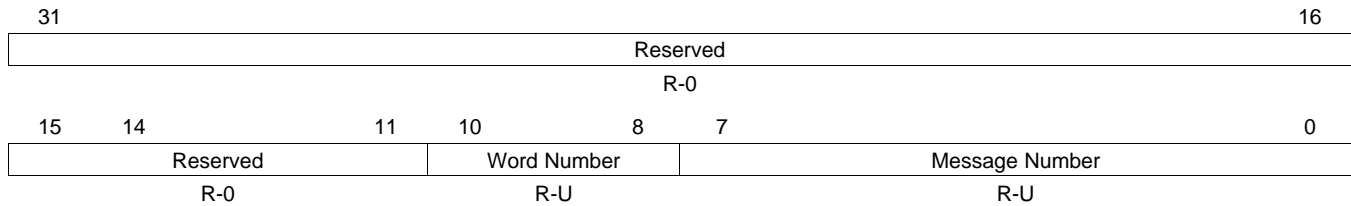


---

**NOTE:** When the internal loop back mode is active (bit LBack is set), bit EXL will be ignored.

---

### 22.18.7 Parity Error Code Register (DCAN PERR)

**Figure 22-26. Parity Error Code Register (DCAN PERR) [offset = 1Ch]**


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

**Table 22-13. Parity Error Code Register Field Descriptions**

Bit	Field	Value	Description
31-11	Reserved	0	These bits are always read as 0. Writes have no effect.
10-8	Word Number	1h-5h	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 <a href="#">Section 22.6.4</a> ).
7-0	Message Number	1h-80h	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.

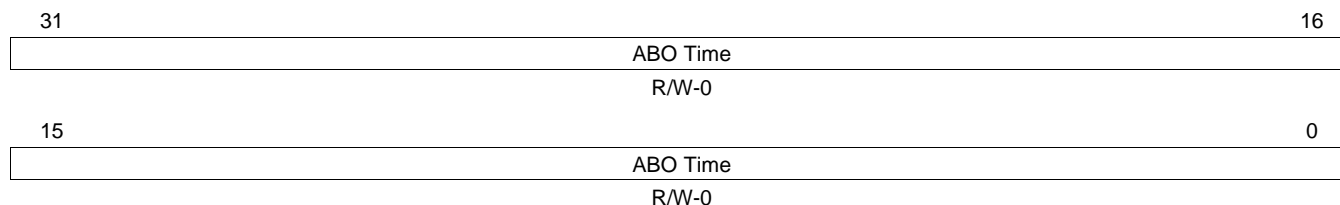
In addition to the PER flag, the Parity Error Code Register will indicate the memory area where the parity error has been detected (message number and word number).

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.

## 22.18.8 Auto-Bus-On Time Register (DCAN ABOTR)

Figure 22-27. Auto-Bus-On Time Register (DCAN ABOTR) [offset = 80h]



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

Table 22-14. Auto-Bus-On Time Register Field Descriptions

Bit	Field	Value	Description
31-0	ABO Time	0-FFFF FFFFh	<p>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.</p> <p>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.</p> <p>The counter will be reloaded with the preload value of the ABO Time register after this phase.</p>

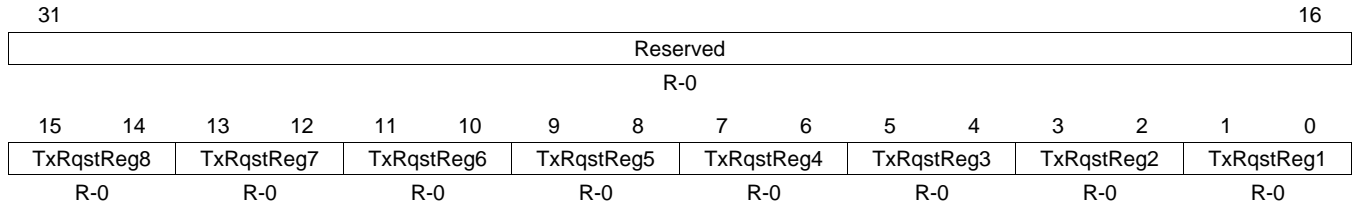
**NOTE:** On write access to the CAN Control register while Auto-Bus-On timer is running, the Auto-Bus-On procedure will be aborted.

**NOTE:** During Debug/Suspend mode, running Auto-Bus-On timer will be paused.

**22.18.9 Transmission Request X Register (DCAN TXRQ X)**

With the Transmission Request X Register, the CPU can detect if one or more bits in the different Transmission Request Registers are set. Each register bit represents a group of eight message objects. If at least one of the TxRqst bits of these message objects are set, the corresponding bit in the Transmission Request X Register will be set.

**Figure 22-28. Transmission Request X Register (DCAN TXRQ X) [offset = 84h]**



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

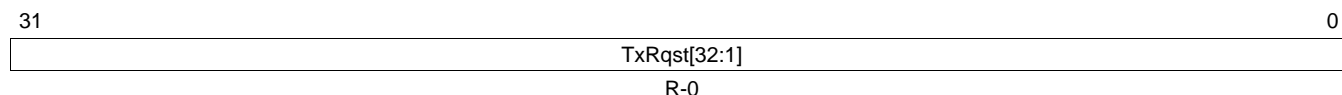
**Example 1**

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.

### 22.18.10 Transmission Request Registers (DCAN TXRQ12 to DCAN TXRQ78)

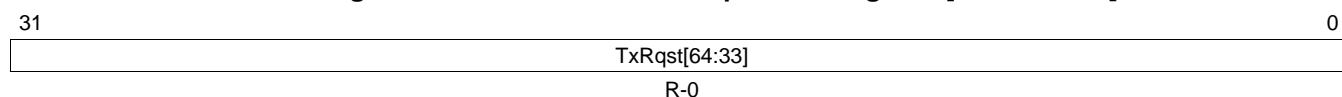
These registers hold the TxRqst bits of the implemented message objects. By reading out these bits, the CPU can check for pending transmission requests. The TxRqst bit in a specific message object can be set/reset by the CPU via the IF1/IF2 Message Interface Registers, or by the Message Handler after reception of a remote frame or after a successful transmission.

**Figure 22-29. Transmission Request 12 Register [offset = 88h]**



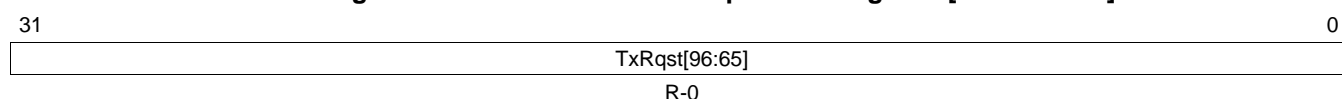
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Figure 22-30. Transmission Request 34 Register [offset = 8Ch]**



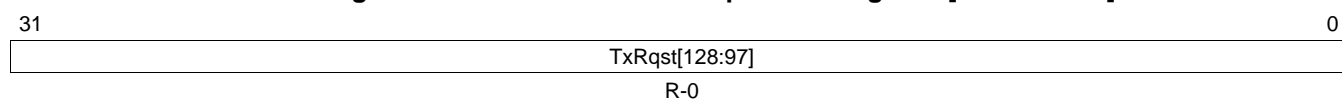
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Figure 22-31. Transmission Request 56 Register [offset = 90h]**



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

**Figure 22-32. Transmission Request 78 Register [offset = 94h]**



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

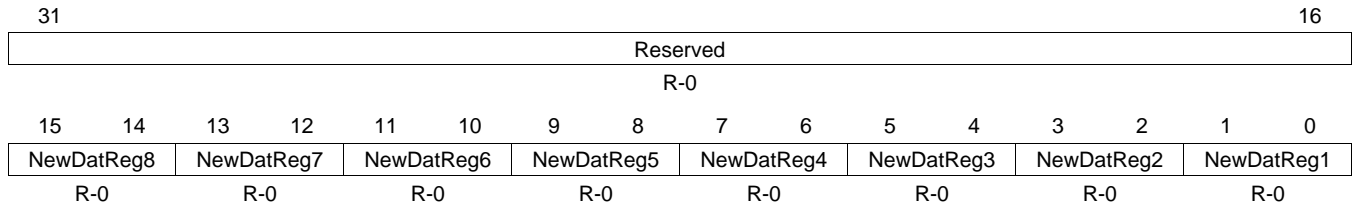
**Table 22-15. Transmission Request Registers Field Descriptions**

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.

**22.18.11 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 one of the NewDat bits of these message objects are set, the corresponding bit in the New Data X Register will be set.

**Figure 22-33. New Data X Register (DCAN NWDAT X) [offset = 98h]**



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

**Equation 1**

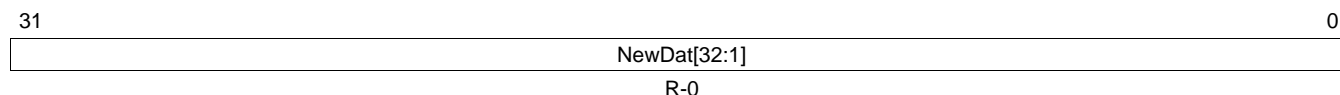
Bit 0 of the New Data X Register represents byte 0 of the New Data 1 Register. If one or more bits in this byte are set, bit 0 of the New Data X Register will be set.



### 22.18.12 New Data Registers (DCAN NWDAT12 to DCAN NWDAT78)

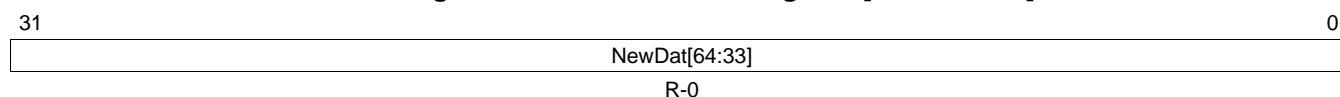
These registers hold the NewDat bits of the implemented message objects. By reading out these bits, the CPU can check for new data in the message objects. The NewDat bit of a specific message object can be set/reset by the CPU via the IF1/IF2 Interface Register sets, or by the Message Handler after reception of a data frame or after a successful transmission.

**Figure 22-34. New Data 12 Register [offset = 9Ch]**



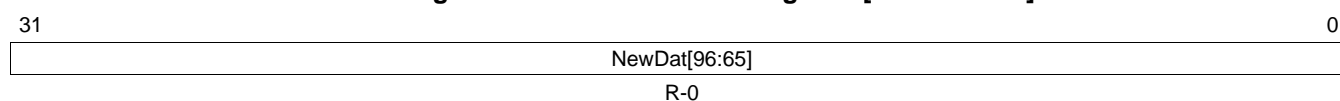
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Figure 22-35. New Data 34 Register [offset = A0h]**



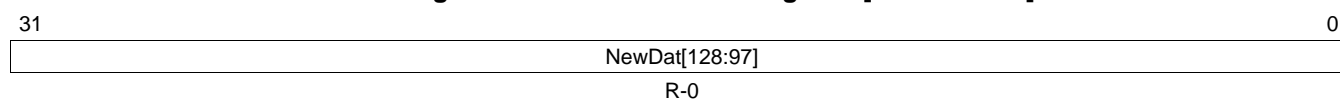
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Figure 22-36. New Data 56 Register [offset = A4h]**



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

**Figure 22-37. New Data 78 Register [offset = A8h]**



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

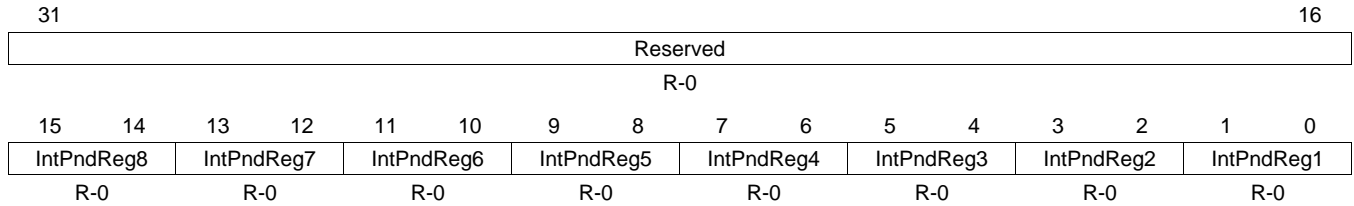
**Table 22-16. New Data Registers Field Descriptions**

Bit	Name	Value	Description
31-0	NewDat[128:1]	0	New Data Bits (for all message objects)
		0	No new data has been written into the data portion of this message object by the Message Handler since the last time when this flag was cleared by the CPU.
		1	The Message Handler or the CPU has written new data into the data portion of this message object.

**22.18.13 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 22-38. Interrupt Pending X Register (DCAN INTPND X) [offset = ACh]**



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

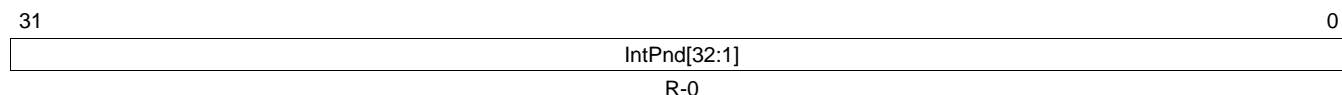
**Example 2**

Bit 0 of the Interrupt Pending X Register represents byte 0 of the Interrupt Pending 1 Register. If one or more bits in this byte are set, bit 0 of the Interrupt Pending X Register will be set.

### 22.18.14 Interrupt Pending Registers (DCAN INTPND12 to DCAN INTPND78)

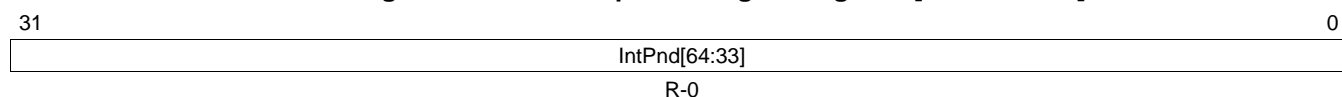
These registers hold the IntPnd bits of the implemented message objects. By reading out these bits, the CPU can check for pending interrupts in the message objects. The IntPnd bit of a specific message object can be set/reset by the CPU via the IF1/IF2 Interface Register sets, or by the Message Handler after a reception or a successful transmission.

**Figure 22-39. Interrupt Pending 12 Register [offset = B0h]**



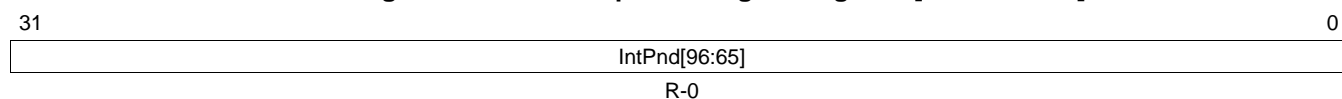
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Figure 22-40. Interrupt Pending 34 Register [offset = B4h]**



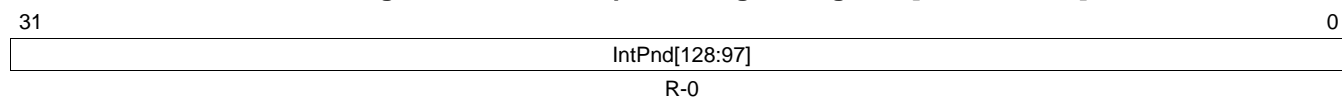
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Figure 22-41. Interrupt Pending 56 Register [offset = B8h]**



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

**Figure 22-42. Interrupt Pending 78 Register [offset = BCh]**



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

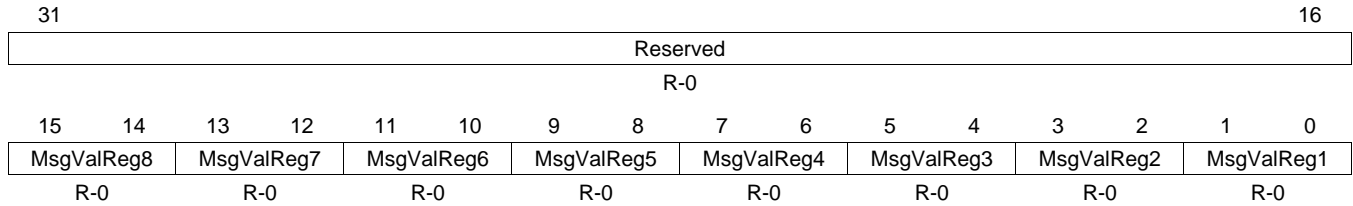
**Table 22-17. Interrupt Pending Registers Field Descriptions**

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.

**22.18.15 Message Valid X Register (DCAN MSGVAL X)**

With the Message Valid X Register, the CPU can detect if one or more bits in the different Message Valid Registers are set. Each bit of this register represents a group of eight message objects. If at least one of the MsgVal bits of these message objects are set, the corresponding bit in the Message Valid X Register will be set.

**Figure 22-43. Message Valid X Register (DCAN MSGVAL X) [offset = C0h]**



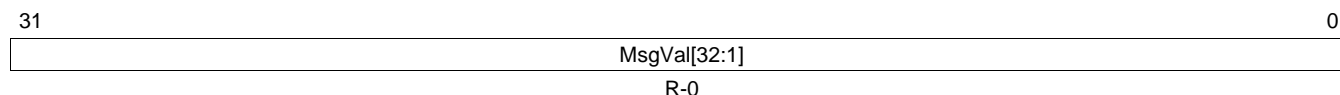
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

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.

### 22.18.16 Message Valid Registers (DCAN MSGVAL12 to DCAN MSGVAL78)

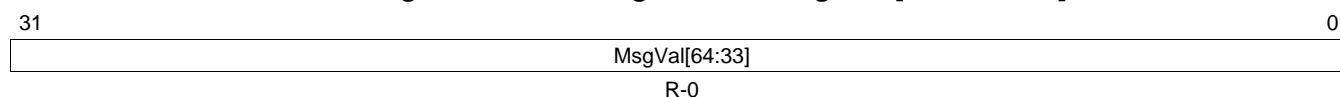
These registers hold the MsgVal bits of the implemented message objects. By reading out these bits, the CPU can check which message objects are valid. The MsgVal bit of a specific message object can be set/reset by the CPU via the IF1/IF2 Interface Register sets, or by the Message Handler after a reception or a successful transmission.

**Figure 22-44. Message Valid 12 Register [offset = C4h]**



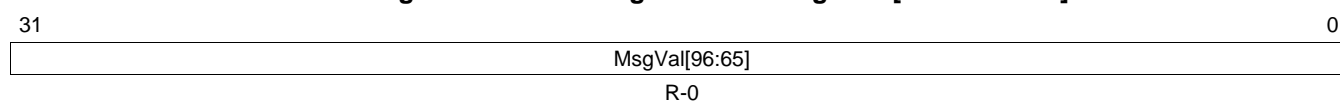
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Figure 22-45. Message Valid 34 Register [offset = C8h]**



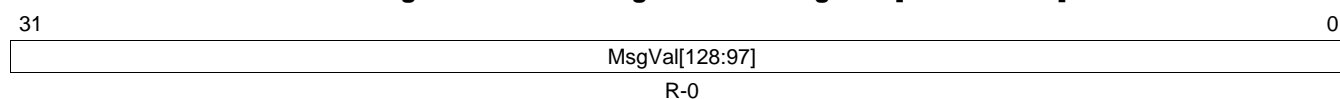
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Figure 22-46. Message Valid 56 Register [offset = CCh]**



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

**Figure 22-47. Message Valid 78 Register [offset = D0h]**



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

**Table 22-18. Message Valid Registers Field Descriptions**

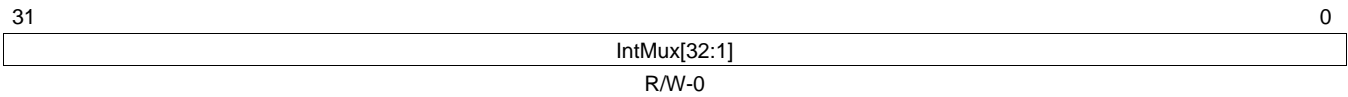
Bit	Name	Value	Description
31-0	MsgVal[128:1]		Message Valid Bits (for all message objects)
		0	This message object is ignored by the Message Handler.
		1	This message object is configured and will be considered by the Message Handler.

**22.18.17 Interrupt 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.

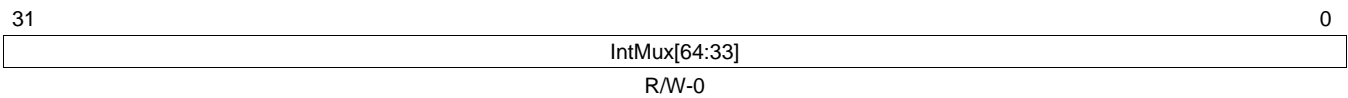
The IntPnd bit of a specific message object can be set or reset by the CPU via the IF1/IF2 Interface Register sets, or by Message Handler after reception or successful transmission of a frame. This will also affect the Int0ID resp Int1ID flags in the Interrupt Register.

**Figure 22-48. Interrupt Multiplexer 12 Register [offset = D8h]**



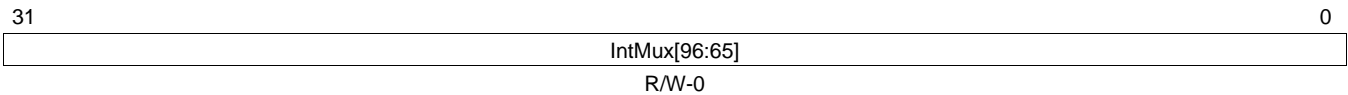
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Figure 22-49. Interrupt Multiplexer 34 Register [offset = DCh]**



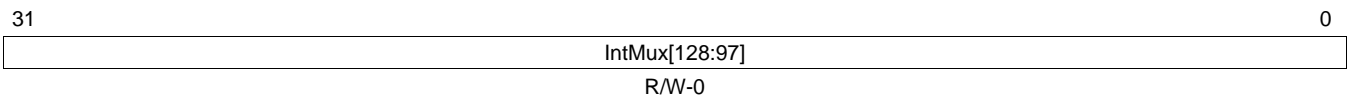
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Figure 22-50. Interrupt Multiplexer 56 Register [offset = E0h]**



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

**Figure 22-51. Interrupt Multiplexer 78 Register [offset = E4h]**



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

**Table 22-19. Interrupt Multiplexer Registers Field Descriptions**

Bit	Name	Value	Description
31-0	IntMux[127:0]		Multiplexes IntPnd value to either DCAN0INT or DCAN1INT interrupt lines. The mapping from the bits to the message objects is as follows: Bit 0 -> last implemented message object. Bit 1 -> message object number 1 Bit 2 -> message object number 2
		0	DCAN0INT line is active if corresponding IntPnd flag is one.
		1	DCAN1INT line is active if corresponding IntPnd flag is one.

### 22.18.18 IF1/IF2 Command Registers (DCAN IF1CMD, DCAN IF2CMD)

The IF1/IF2 Command Register configure and Initiate the transfer between the IF1/IF2 Register sets and the Message RAM. It is configurable which portions of the message object should be transferred.

A transfer is started when the CPU writes the message number to bits [7:0] of the IF1/IF2 Command Register. With this write operation, the Busy bit is automatically set to '1' to indicate that a transfer is in progress.

After 4 to 14 VBUS clock cycles, the transfer between the Interface Register and the Message RAM will be completed and the Busy bit is cleared. The maximum number of cycles is needed when the message transfer concurs with a CAN message transmission, acceptance filtering, or message storage.

If the CPU writes to both IF1/IF2 Command Registers consecutively (request of a second transfer while first transfer is still in progress), the second transfer will start after the first one has been completed.

---

**NOTE:** While 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 22-52. IF1 Command Registers (DCAN IF1CMD) [offset = 100h]**

Reserved							
R-0							
31	24						
23	22	21	20	19	18	17	16
WR/RD	Mask	Arb	Control	ClrIntPnd	TxRqst/NewDat	Data A	Data B
R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0
15	14	13					8
Busy	DMA Active	Reserved					
R-0	R/WP/C-0	R-0					
7							0
Message Number							
R/WP-1h							

LEGEND: R = Read; WP = Write Protected (Protected by Busy bit); C = Clear by IF1 Access; -n = value after reset

**Figure 22-53. IF1 Command Registers (CAN IF2CMD) [offset = 120h]**

31								24							
Reserved															
R-0															
23		22		21		20		19		18		17		16	
WR/RD	Mask	Arb	Control	ClrIntPnd	TxRqst/NewDat	Data A	Data B								
R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0								
15		14		13		Reserved								8	
Busy	DMA Active													8	
R-0	R/WP/C-0	R-0													
7		Message Number												0	
R/WP-1h															

LEGEND: R = Read; WP = Write Protected (Protected by Busy bit); C = Clear by IF1 Access; -n = value after reset

**Table 22-20. IF1/IF2 Command Register Field Descriptions**

Bit	Field	Value	Description
31-24	Reserved		These bits are always read as 0. Writes have no effect.
23	WR/RD	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	0	Access Mask bits 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	0	Access Arbitration bits 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	0	Access Control bits Control bits will not be changed
		1	Direction = Read: The Message Control bits will be transferred from the message object addressed by Message Number (Bits [7:0]) to the corresponding IF1/IF2 Register set Direction = Write: The Message Control bits will be transferred from the IF1/IF2 Register set to the message object addressed by Message Number (Bits [7:0]) If the TxRqst/NewDat bit in this register (Bit [18]) is set, the TxRqst/NewDat bit in the IF1/IF2 Message Control Register will be ignored
19	ClrIntPnd	0	Clear Interrupt Pending bit IntPnd bit will not be changed
		1	Direction = Read: Clears IntPnd bit in the message object Direction = Write: This bit is ignored. Copying of IntPnd flag from IF1/IF2 Registers to Message RAM can be controlled by only the Control flag (Bit [20])



**Table 22-20. IF1/IF2 Command Register Field Descriptions (continued)**

Bit	Field	Value	Description
18	TxRqst/NewDat	0 1	<p>Access Transmission Request bit</p> <p>Direction = Read: NewDat bit will not be changed. Direction = Write: TxRqst/NewDat bit will be handled according to the Control bit</p> <p>Direction = Read: Clears NewDat bit in the message object. Direction = Write: Sets TxRqst/NewDat in the message object</p> <p><b>Note:</b> If a CAN transmission is requested by setting TxRqst/NewDat in this register, the TxRqst/NewDat bits in the message object will be set to '1' and independent of the values in IF1/IF2 Message Control Register. A read access to a message object can be combined with the reset of the control bits IntPnd and NewDat. The values of these bits transferred to the IF1/IF2 Message Control Register always reflect the status before resetting them.</p>
17	Data A	0 1	<p>Access Data Bytes 0–3</p> <p>Data Bytes 0–3 will not be changed</p> <p>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])</p> <p><b>Note:</b> The duration of the message transfer is independent of the number of bytes to be transferred.</p>
16	Data B	0 1	<p>Access Data Bytes 4–7</p> <p>Data Bytes 4–7 will not be changed</p> <p>Direction = Read: The Data Bytes 4–7 will be transferred from the message object addressed by the Message Number (Bits [7:0]) to the corresponding IF1/IF2 Register set Direction = Write: The Data Bytes 4–7 will be transferred from the IF1/IF2 Register set to the message object addressed by the Message Number (Bits [7:0])</p> <p><b>Note:</b> The duration of the message transfer is independent of the number of bytes to be transferred.</p>
15	Busy	0 1	<p>Busy flag</p> <p>0 No transfer between IF1/IF2 Register set and Message RAM is in progress 1 Transfer between IF1/IF2 Register set and Message RAM is in progress. This bit is set to 1 after the message number has been written to bits [7:0]. IF1/IF2 Register set will be write-protected. The bit is cleared after read/write action has finished.</p>
14	DMA Active	0 1	<p>Activation of DMA feature for subsequent internal IF1/IF2 update</p> <p>0 DMA request line is independent of IF1/IF2 activities 1 DMA is requested after completed transfer between IF1/IF2 Register set and Message RAM. The DMA request remains active until the first read or write to one of the IF1/IF2 registers. An exception is a write to Message Number (Bits [7:0]) when DMA Active is 1.</p> <p><b>Note:</b> Due to the auto reset feature of the DMA Active bit, this bit has to be separately set for each subsequent DMA cycle.</p>
13-8	Reserved		These bits are always read as 0. Writes have no effect.
7-0	Message Number	0x00 0x01- 0x80 0x81- 0xFF	<p>Number of message object in Message RAM that is used for data transfer</p> <p>Invalid message number</p> <p>Valid message numbers if 128 message objects are implemented</p> <p>Invalid message numbers if 128 message objects are implemented</p> <p><b>Note:</b> When an invalid message number is written to the IF1/IF2 Command Register which is higher than the last implemented message object number, a modulo addressing will occur. When e.g. accessing message object 33 in a DCAN module with 32 message objects only, the message object 1 will be accessed instead.</p>

**22.18.19 IF1/IF2 Mask Registers (DCAN IF1MSK, DCAN IF2MSK)**

The bits of the IF1/IF2 Mask Registers mirror the mask bits of a message object. The function of the relevant message objects bits is described in [Section 22.6.1](#).

**NOTE:** While Busy bit of IF1/IF2 Command Register is one, IF1/IF2 Register Set is write protected.

**Figure 22-54. IF1 Mask Register (DCAN IF1MSK) [offset = 104h]**

31	30	29	28	16
MXtd	MDir	Rsvd.	Msk[28:16]	
R/WP- 1	R/WP- 1	R-1	R/WP-1FFFh	
15				0
Msk[15:0]				
R/WP-FFFFh				

LEGEND: R = Read; WP = Write-protected (Protected by Busy bit); -n = value after reset

**Figure 22-55. IF2 Mask Register (DCAN IF2MSK) [offset = 124h]**

31	30	29	28	16
MXtd	MDir	Rsvd.	Msk[28:16]	
R/WP- 1	R/WP- 1	R-1	R/WP-1FFFh	
15				0
Msk[15:0]				
R/WP-FFFFh				

LEGEND: R = Read; WP = Write-protected (Protected by Busy bit); -n = value after reset

**Table 22-21. IF1/IF2 Mask Register Field Descriptions**

Bit	Field	Value	Description
31	MXtd	0	Mask Extended Identifier The extended identifier bit (IDE) has no effect on the acceptance filtering.
		1	The extended identifier bit (IDE) is used for acceptance filtering.  When 11-bit ("standard") identifiers are used for a message object, the identifiers of received Data Frames are written into bits ID[28:18]. For acceptance filtering, only these bits with mask bits Msk[28:18] are considered.
30	MDir	0	Mask Message Direction 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]	0	Identifier Mask 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

### 22.18.20 IF1/IF2 Arbitration Registers (DCAN IF1ARB, DCAN IF2ARB)

The bits of the IF1/IF2 Arbitration Registers mirror the arbitration bits of a message object. The function of the relevant message objects bits is described in [Section 22.6.1](#).

**NOTE:** While Busy bit of IF1/IF2 Command Register is one, IF1/IF2 Register Set is write protected.

**Figure 22-56. IF1 Arbitration Register (DCAN IF1ARB) [offset = 108h]**

31	30	29	28	16
MsgVal	Xtd	Dir	ID[28:16]	
R/WP-0	R/W P-0	R/WP- 0	R/WP-0	
15			ID[15:0]	0
			R/WP-0	

LEGEND: R = Read; WP = Write-protected (Protected by Busy bit); -n = value after reset

**Figure 22-57. IF2 Arbitration Register (DCAN IF2ARB) [offset = 128h]**

31	30	29	28	16
MsgVal	Xtd	Dir	ID[28:16]	
R/WP-0	R/W P-0	R/WP- 0	R/WP-0	
15			ID[15:0]	0
			R/WP-0	

LEGEND: R = Read; WP = Write-protected (Protected by Busy bit); -n = value after reset

**Table 22-22. IF1/IF2 Arbitration Register Field Descriptions**

Bit	Field	Value	Description
31	MsgVal	0 1	Message Valid The message object is ignored by the Message Handler. The message object is used by the Message Handler. Note: The CPU should reset the MsgVal bit of all unused Messages Objects during the initialization before it resets bit Init in the CAN Control Register. MsgVal must also be reset if the messages object is no longer used in operation. For reconfiguration of message objects during normal operation, see <a href="#">Section 22.8.6</a> and <a href="#">Section 22.8.7</a> .
30	Xtd	0 1	Extended Identifier The 11-bit ("standard") identifier is used for this message object. The 29-bit ("extended") identifier is used for this message object.
29	Dir	0 1	Message direction 0 Direction = Receive: On TxRqst, a Remote Frame with the identifier of this message object is transmitted. On receiving a Data Frame with a matching identifier, this message is stored in this message object. 1 Direction = Transmit: On TxRqst, the respective message object is transmitted as a Data Frame. On receiving a Remote Frame with a matching identifier, the TxRqst bit of this message object is set (if RmtEn = 1).
28-0	ID[28:0]	ID[28:0] ID[28:18]	Message Identifier 29-bit Identifier ("Extended Frame") 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.

### 22.18.21 IF1/IF2 Message Control Registers (DCAN IF1MCTL, DCAN IF2MCTL)

The bits of the IF1/IF2 Message Control Registers mirror the message control bits of a message object. The function of the relevant message objects bits is described in [Section 22.6.1](#).

**NOTE:** While Busy bit of IF1/IF2 Command Register is one, IF1/IF2 Register Set is write protected.

**Figure 22-58. IF1 Message Control Register (DCAN IF1MCTL) [offset = 10Ch]**

Reserved							
R-0							
15	14	13	12	11	10	9	8
NewDat	MsgLst	IntPnd	UMask	TxIE	RxIE	RmtEn	TxRqst
R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0
7	6	4		3			
EoB	Reserved			DLC[3:0]			
R/WP-0	R-0			R/WP-0			

LEGEND: R = Read; WP = Write-protected (Protected by Busy bit); -n = value after reset

**Figure 22-59. IF2 Message Control Register (DCAN IF2MCTL) [offset = 12Ch]**

Reserved							
R-0							
15	14	13	12	11	10	9	8
NewDat	MsgLst	IntPnd	UMask	TxIE	RxIE	RmtEn	TxRqst
R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0
7	6	4		3			
EoB	Reserved			DLC[3:0]			
R/WP-0	R-0			R/WP-0			

LEGEND: R = Read; WP = Write-protected (Protected by Busy bit); -n = value after reset

**Table 22-23. IF1/IF2 Message Control Register Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved		These bits are always read as 0. Writes have no effect.
15	NewDat	0	No new data has been written into the data portion of this message object by the Message Handler since the last time this flag was cleared by the CPU.
		1	The Message Handler or the CPU has written new data into the data portion of this message object.
14	MsgLst	0	Message Lost (only valid for message objects with direction = receive) 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	0	Interrupt Pending 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.

**Table 22-23. IF1/IF2 Message Control Register Field Descriptions (continued)**

Bit	Field	Value	Description
12	UMask	0 1	Use Acceptance Mask Mask ignored Use Mask (Msk[28:0], MXtd, and MDir) for acceptance filtering. If the UMask bit is set to 1, the message object's mask bits have to be programmed during initialization of the message object before MsgVal is set to 1.
11	TxE	0 1	Transmit Interrupt Enable IntPnd will not be triggered after the successful transmission of a frame. IntPnd will be triggered after the successful transmission of a frame.
10	RxE	0 1	Receive Interrupt Enable IntPnd will not be triggered after the successful transmission of a frame. IntPnd will be triggered after the successful transmission of a frame.
9	RmtEn	0 1	Remote Enable At the reception of a Remote Frame, TxRqst is not changed. At the reception of a Remote Frame, TxRqst is set.
8	TxRqst	0 1	Transmit Request This message object is not waiting for a transmission. The transmission of this message object is requested and not yet done.
7	EoB	0 1	Data Frame has 0–8 data bits Data Frame has 8 data bytes <b>Note:</b> 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]	0-8 9-15	Data Length Code Data Frame has 0–8 data bits. Data Frame has 8 data bytes. <b>Note:</b> 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.

### 22.18.22 IF1/IF2 Data A and Data B Registers (DCAN IF1DATA/DATB, DCAN IF2DATA/DATB)

The data bytes of CAN messages are stored in the IF1/IF2 registers in the following order.

In a CAN Data Frame, Data 0 is the first, and Data 7 is the last byte to be transmitted or received. In CAN's serial bit stream, the MSB of each byte will be transmitted first

**Figure 22-60. IF1 Data A Register (DCAN IF1DATA) [offset = 110h]**

31	24	23	16
Data 3			Data 2
R/WP-0			R/WP-0
15	8	7	0
Data 1			Data 0
R/WP-0			R/WP-0

LEGEND: R = Read; WP = Write-protected (Protected by Busy bit); -n = value after reset

**Figure 22-61. IF1 Data B Register (DCAN IF1DATB) [offset = 114h]**

31	24	23	16
Data 7			Data 6
R/WP-0			R/WP-0
15	8	7	0
Data 5			Data 4
R/WP-0			R/WP-0

LEGEND: R = Read; WP = Write-protected (Protected by Busy bit); -n = value after reset

**Figure 22-62. IF2 Data A Register (DCAN IF2DATA) [offset = 130h]**

31	24	23	16
Data 7			Data 6
R/WP-0			R/WP-0
15	8	7	0
Data 5			Data 4
R/WP-0			R/WP-0

LEGEND: R = Read; WP = Write-protected (Protected by Busy bit); -n = value after reset

**Figure 22-63. IF2 Data B Register (DCAN IF2DATB) [offset = 134h]**

31	24	23	16
Data 7			Data 6
R/WP-0			R/WP-0
15	8	7	0
Data 5			Data 4
R/WP-0			R/WP-0

LEGEND: R = Read; WP = Write-protected (Protected by Busy bit); -n = value after reset

**22.18.23 IF3 Observation Register (DCAN IF3OBS)**

The IF3 register set can automatically be updated with received message objects without the need to initiate the transfer from Message RAM by CPU (Additional information can be found in [Section 22.6.1](#)).

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.

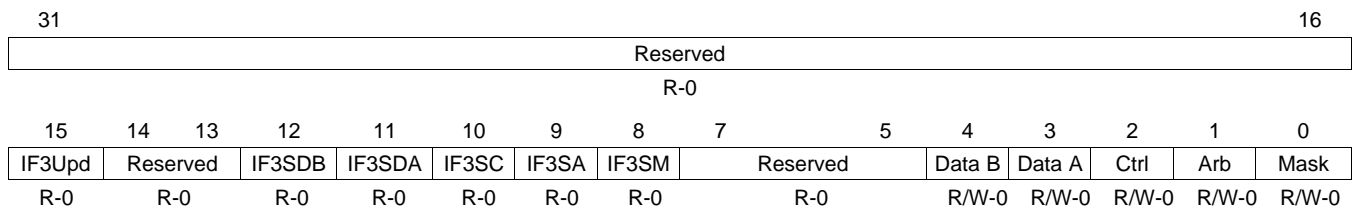
A write access to this register aborts a pending DMA cycle by resetting the DMA line and enables updating of IF3 Interface Register set with new data. To avoid data inconsistency, the DMA controller should be disabled before reconfiguring IF3 observation register.

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. See the device data sheet to find out if this interrupt source is available

With this, the observation status bits and the IF3Upd bit could be used by the application to realize the notification about new IF3 content in polling or interrupt mode.

**Figure 22-64. IF3 Observation Register (DCAN IF3OBS) [offset = 140h]**



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

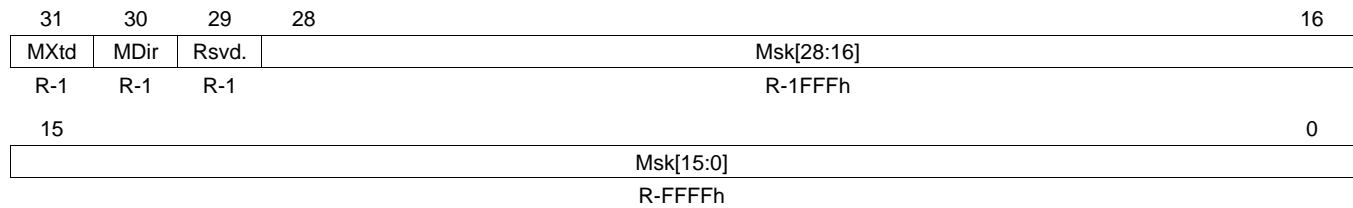
**Table 22-24. IF3 Observation Register Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved		These bits are always read as 0. Writes have no effect.
15	IF3 Upd	0 1	IF3 Update Data No new data has been loaded since IF3 was last read New data has been loaded since IF3 was last read
14-13	Reserved		These bits are always read as 0. Writes have no effect
12	IF3 SDB	0 1	IF3 Status of Data B read access All Data B bytes are already read or are not marked to be read Data B section still has data to read
11	IF3 SDA	0 1	IF3 Status of Data A read access All Data A bytes are already read or are not marked to be read Data A section still has data to read
10	IF3 SC	0 1	IF3 Status of Control bits read access All Control section bytes are already read or are not marked to be read Control section still has data to read



**Table 22-24. IF3 Observation Register Field Descriptions (continued)**

Bit	Field	Value	Description
9	IF3 SA		IF3 Status of Arbitration data read access
		0	All Arbitration data bytes are already read or are not marked to be read
		1	Arbitration section still has data to read
8	IF3 SM		IF3 Status of Mask data read access
		0	All Mask data bytes are already read or are not marked to be read
		1	Mask section still has data to read
7-5	Reserved		These bits are always read as 0. Writes have no effect
4	Data B		Data B read observation
		0	Data B section does not need to be read
		1	Data B section has to be read to enable next IF3 update
3	Data A		Data A read observation
		0	Data A section does not need to be read
		1	Data A section has to be read to enable next IF3 update
2	Ctrl		Ctrl read observation
		0	Ctrl section does not need to be read
		1	Ctrl section has to be read to enable next IF3 update
1	Arb		Arbitration data read observation
		0	Arbitration data does not need to be read
		1	Arbitration data has to be read to enable next IF3 update
0	Mask		Mask data read observation
		0	Mask data does not need to be read
		1	Mask data has to be read to enable next IF3 update

**22.18.24 IF3 Mask Register (DCAN IF3MSK)**
**Figure 22-65. IF3 Mask Register (DCAN IF3MSK) [offset = 144h]**


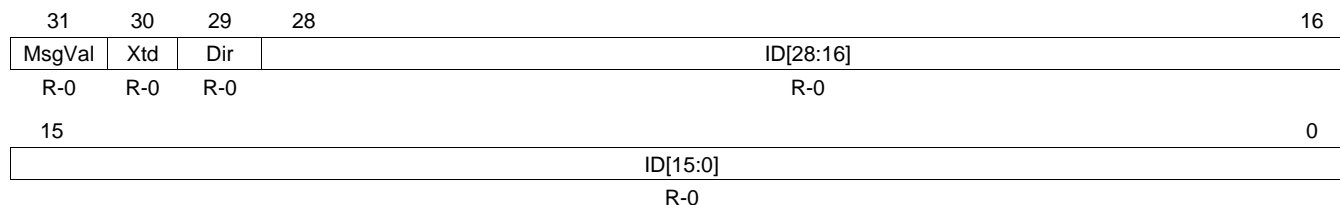
LEGEND: R = Read; -n = value after reset

**Table 22-25. IF3 Mask Register Field Descriptions**

Bit	Field	Value	Description
31	MXtd	0 1	Mask Extended Identifier The extended identifier bit (IDE) has no effect on acceptance filtering. The extended identifier bit (IDE) is used for acceptance filtering. <b>Note:</b> 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	0 1	Mask Message Direction The message direction bit (Dir) has no effect on acceptance filtering. 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]	0 1	Identifier Mask The corresponding bit in the identifier of the message object is not used for acceptance filtering (don't care). The corresponding bit in the identifier of the message object is used for acceptance filtering.

### 22.18.25 IF3 Arbitration Register (DCAN IF3ARB)

**Figure 22-66. IF3 Arbitration Register (DCAN IF3ARB) [offset = 148h]**



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

**Table 22-26. IF3 Arbitration Register Field Descriptions**

Bit	Field	Value	Description
31	MsgVal	0 1	Message Valid 0 The message object is ignored by the Message Handler. 1 The message object is to be used by the Message Handler. <b>Note:</b> The CPU should reset the MsgVal bit of all unused Messages Objects during the initialization before it resets bit Init in the CAN Control Register. MsgVal must also be reset if the messages object is no longer used in operation. For reconfiguration of message objects during normal operation, see <a href="#">Section 22.8.6</a> and <a href="#">Section 22.8.7</a> .
30	Xtd	0 1	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	0 1	Message direction 0 Direction = Receive: On TxRqst, a remote frame with the identifier of this message object is transmitted. On receiving a data frame with a matching identifier, the message is stored in this message object. 1 Direction = Transmit: On TxRqst, the respective message object is transmitted as a data frame. On receiving a remote frame with a matching identifier, the TxRqst bit of this message object is set (if RmtEn = one).
28-0	ID[28:0]	ID[28:0] ID[28:18]	Message Identifier ID[28:0] 29-bit Identifier ("Extended Frame") ID[28:18] 11-bit Identifier ("Standard Frame")

**22.18.26 IF3 Message Control Register (DCAN IF3MCTL)**
**Figure 22-67. IF3 Message Control Register (DCAN IF3MCTL) [offset = 14Ch]**

31								16							
Reserved															
R-0															
15		14		13		12		11		10		9		8	
NewDat	MsgLst	IntPnd	UMask	TxIE	RxIE	RmtEn	TxRqst								
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0								
7		6		4		3		0							
EoB		Reserved						DLC[3:0]							
R-0		R-0						R-0							

LEGEND: R = Read; WP = Write-protected (Protected by Busy bit); -n = value after reset

**Table 22-27. IF3 Message Control Register Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved		These bits are always read as 0. Writes have no effect.
15	NewDat	0	No new data has been written into the data portion of this message object by the Message Handler since the last time this flag was cleared by the CPU.
		1	The Message Handler or the CPU has written new data into the data portion of this message object.
14	MsgLst	0	Message Lost (only valid for message objects with direction = receive) 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	0	Interrupt Pending 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	0	Use Acceptance Mask Mask ignored
		1	Use Mask (Msk[28:0], MXtd, and MDir) for acceptance filtering. If the UMask bit is set to 1, the message object's mask bits have to be programmed during initialization of the message object before MsgVal is set to 1.
11	TxIE	0	Transmit Interrupt Enable 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	0	Receive Interrupt Enable IntPnd will not be triggered after the successful transmission of a frame.
		1	IntPnd will be triggered after the successful transmission of a frame.
9	RmtEn	0	Remote Enable At the reception of a Remote Frame, TxRqst is not changed.
		1	At the reception of a Remote Frame, TxRqst is set.
8	TxRqst	0	Transmit Request This message object is not waiting for a transmission.
		1	The transmission of this message object is requested and not yet done.

**Table 22-27. IF3 Message Control Register Field Descriptions (continued)**

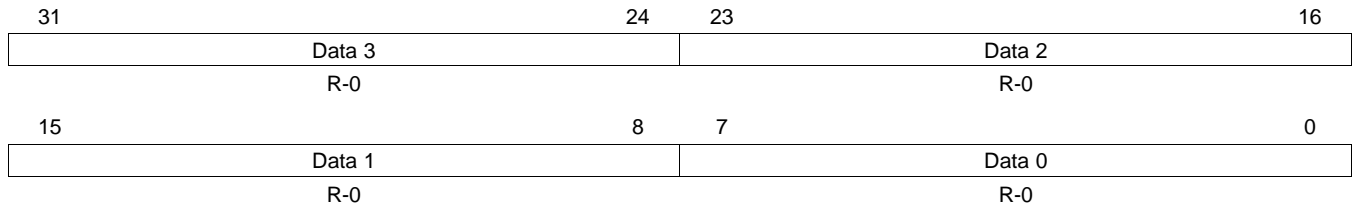
Bit	Field	Value	Description
7	EoB	0 1	<p>End of Block</p> <p>The message object is part of a FIFO Buffer block and is not the last message object of the FIFO Buffer block</p> <p>The message object is a single message object or the last message object in a FIFO Buffer block</p> <p><b>Note:</b> This bit 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.</p>
6-4	Reserved		These bits are always read as 0. Writes have no effect.
3-0	DLC[3:0]	0-8 9-15	<p>Data Length Code</p> <p>Data Frame has 0–8 data bits.</p> <p>Data Frame has 8 data bytes.</p> <p><b>Note:</b> 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.</p>

**22.18.27 IF3 Data A and Data B Registers (DCAN IF3DATA/DATB)**

The data bytes of CAN messages are stored in the IF3 registers in the following order.

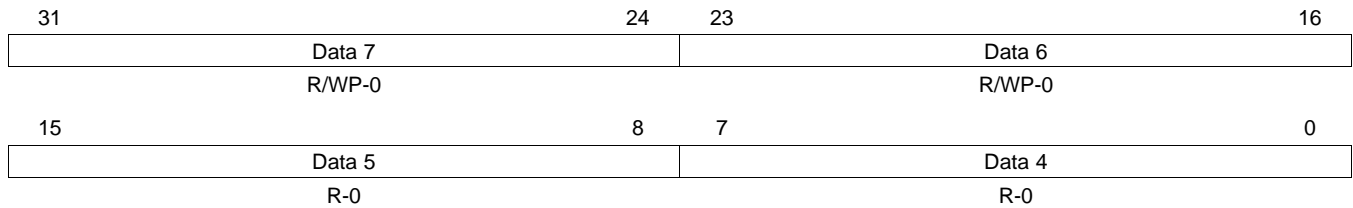
In a CAN Data Frame, Data 0 is the first, and Data 7 is the last byte to be transmitted or received. In CAN's serial bit stream, the MSB of each byte will be transmitted first.

**Figure 22-68. IF3 Data A Register (DCAN IF3DATA) [offset = 150h]**



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

**Figure 22-69. IF3 Data B Register (DCAN IF3DATB) [offset = 154h]**



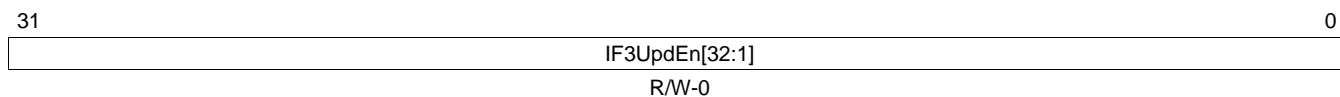
LEGEND: R = Read; -n = value after reset

### 22.18.28 IF3 Update Enable Registers (DCAN IF3UPD12 to IF3UPD78)

The automatic update functionality of the IF3 register set can be configured for each message object. A message object is enabled for automatic IF3 update, if the dedicated IF3UpdEn flag is set. This means that an active NewDat flag of this message object (e.g due to reception of a CAN frame) will trigger an automatic copy of the whole message object to IF3 register set.

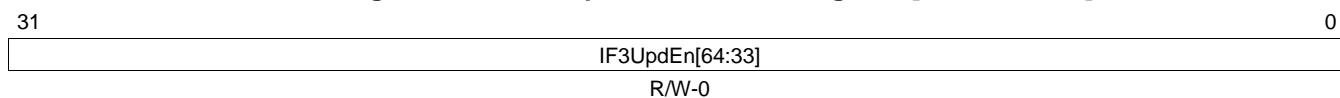
**NOTE:** IF3 Update enable should not be set for transmit objects.

**Figure 22-70. IF3 Update Enable 12 Register [offset = 160h]**



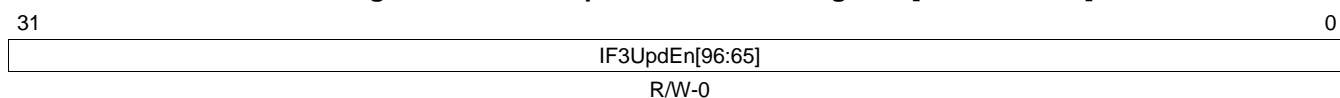
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Figure 22-71. IF3 Update Enable 34 Register [offset = 164h]**



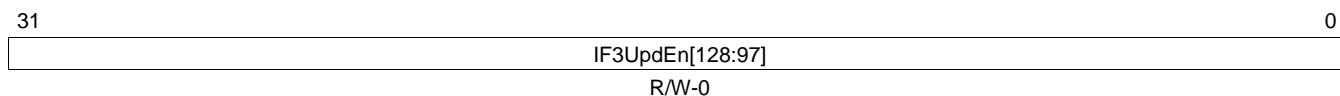
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Figure 22-72. IF3 Update Enable 56 Register [offset = 168h]**



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

**Figure 22-73. IF3 Update Enable 78 Register [offset = 16Ch]**



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

**Table 22-28. IF3 Update Control Register Field Descriptions**

Bit	Name	Value	Description
31-0	IF3UpdEn[128:1]	0	IF3 Update Enabled (for all message objects) 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.

**22.18.29 CAN TX IO Control Register (DCAN TIOC)**

The CAN\_TX pin of the DCAN module can be used as general purpose IO pin if CAN function is not needed.

**NOTE:** The values of the IO Control registers are only writable if Init bit of CAN Control Register is set.

**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 22-74. CAN TX IO Control Register (DCAN TIOC) [offset = 1E0h]**

31	Reserved	19	18	17	16
	R-0		R/W-D	R/W-D	R/WP-0
15	Reserved	4	3	2	1
	R-0		R/WP-0	R/WP-0	R/WP-0
			Func	Dir	Out
			R/WP-0	R/WP-0	R/WP-0
					In
					R-U

LEGEND: R = Read; W = Write; WP = Write-protected (protected by Init bit); D = Device-dependent; -n = value after reset

**Table 22-29. CAN TX IO Control Register Field Descriptions**

Bit	Field	Value	Description
31-19	Reserved		These bits are always read as 0. Writes have no effect.
18	PU	0 1	CAN_TX Pullup/Pulldown select. This bit is only active when CAN_TX is configured to be an input. CAN_TX Pulldown is selected, when pull logic is active (PD = 0). CAN_TX Pullup is selected, when pull logic is active (PD = 0).
17	PD	0 1	CAN_TX pull disable. This bit is only active when CAN_TX is configured to be an input. CAN_TX pull is active. CAN_TX pull is disabled.
16	OD	0 1	CAN_TX open drain enable. This bit is only active when CAN_TX is configured to be in GIO mode (TIOC.Func = 0). The CAN_TX pin is configured in push/pull mode. The CAN_TX pin is configured in open drain mode. Forced to '0' if Init bit of CAN control register is reset.
15-4	Reserved		These bits are always read as 0. Writes have no effect.
3	Func	0 1	CAN_TX function. This bit changes the function of the CAN_TX pin. CAN_TX pin is in GIO mode. CAN_TX pin is in functional mode (as an output to transmit CAN data). Forced to '1' if Init bit of CAN control register is reset.
2	Dir	0 1	CAN_TX data direction. This bit controls the direction of the CAN_TX pin when it is configured to be in GIO mode only (TIOC.Func = 0). The CAN_TX pin is an input. The CAN_TX pin is an output. Forced to '1' if Init bit of CAN control register is reset.



**Table 22-29. CAN TX IO Control Register Field Descriptions (continued)**

Bit	Field	Value	Description
1	Out	0 1	<p>CAN_TX data out write. This bit is only active when CAN_TX pin is configured to be in GIO mode (TIOC.Func = 0) and configured to be an output pin (TIOC.Dir = 1). The value of this bit indicates the value to be output to the CAN_TX pin.</p> <p>The CAN_TX pin is driven to logic low (0).</p> <p>The CAN_TX pin is at logic high (1).</p> <p>Forced to Tx output of the CAN Core, if Init bit of CAN Control register is reset.</p>
0	In	0 1	<p>CAN_TX data in.</p> <p>The CAN_TX pin is at logic low (0).</p> <p>The CAN_TX pin is at logic high (1).</p> <p><b>Note:</b> 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 the DCAN module is reset).</p>

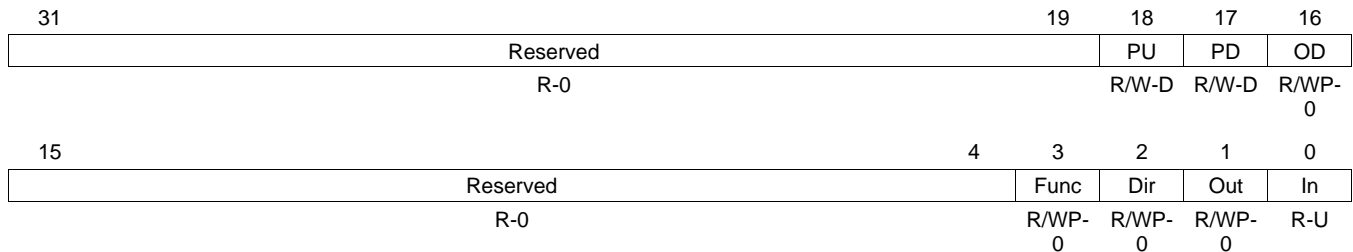
### 22.18.30 CAN RX IO Control Register (DCAN RIOC)

The CAN\_RX pin of the DCAN module can be used as general purpose IO pin if CAN function is not needed.

**NOTE:** The values of the IO Control registers are writable only if Init bit of CAN Control Register is set.

**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 22-75. CAN RX IO Control Register (DCAN RIOC) [offset = 1E4h]**



LEGEND: R = Read; W = Write; WP = Write-protected (protected by Init bit); D = Device-dependent; -n = value after reset

**Table 22-30. CAN RX IO Control Register Field Descriptions**

Bit	Field	Value	Description
31-19	Reserved		These bits are always read as 0. Writes have no effect.
18	PU	0	CAN_RX Pullup/Pulldown select. This bit is only active when CAN_RX is configured to be an input. CAN_RX Pulldown is selected, when pull logic is active (PD = 0).
		1	CAN_RX Pullup is selected, when pull logic is active (PD = 0).
17	PD	0	CAN_RX pull disable. This bit is only active when CAN_RX is configured to be an input. CAN_RX pull is active.
		1	CAN_RX pull is disabled.
16	OD	0	CAN_RX open drain enable. This bit is only active when CAN_RX is configured to be in GIO mode (RIOC.Func = 0). The CAN_RX pin is configured in push/pull mode.
		1	The CAN_RX pin is configured in open drain mode. Forced to '0' if Init bit of CAN control register is reset.
15-4	Reserved		These bits are always read as 0. Writes have no effect.
3	Func	0	CAN_RX function. This bit changes the function of the CAN_RX pin. CAN_RX pin is in GIO mode.
		1	CAN_RX pin is in functional mode (as an output to transmit CAN data). Forced to '1' if Init bit of CAN control register is reset.
2	Dir	0	CAN_RX data direction. This bit controls the direction of the CAN_RX pin when it is configured to be in GIO mode only (RIOC.Func = 0). The CAN_RX pin is an input.
		1	The CAN_RX pin is an output. Forced to '1' if Init bit of CAN control register is reset.
1	Out	0	CAN_RX data out write. This bit is only active when CAN_RX pin is configured to be in GIO mode (RIOC.Func = 0) and configured to be an output pin (RIOC.Dir = 1). The value of this bit indicates the value to be output to the CAN_RX pin. The CAN_RX pin is driven to logic low (0).
		1	The CAN_RX pin is at logic high (1).

**Table 22-30. CAN RX IO Control Register Field Descriptions (continued)**

Bit	Field	Value	Description
0	In		CAN_RX data in.
		0	The CAN_RX pin is at logic low (0).
		1	The CAN_RX pin is at logic high (1).
			<b>Note:</b> 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 the DCAN module is reset).

## **Multi-Buffered Serial Peripheral Interface Module (MibSPI) with Parallel Pin Option (MibSPIP)**

This chapter provides the specifications for a 16-bit configurable synchronous multi-buffered multi-pin serial peripheral interface (MibSPI). This chapter also provides the specifications for MibSPI with Parallel Pin Option (MibSPIP). The MibSPI is a programmable-length shift register used for high-speed communication between external peripherals or other microcontrollers.

Throughout this chapter, all references to SPI also apply to MibSPI/MibSPIP, unless otherwise noted.

Topic	Page
<b>23.1 Overview</b> .....	<b>1234</b>
<b>23.2 Operating Modes</b> .....	<b>1236</b>
<b>23.3 Test Features</b> .....	<b>1255</b>
<b>23.4 General-Purpose I/O</b> .....	<b>1257</b>
<b>23.5 Low-Power Mode</b> .....	<b>1258</b>
<b>23.6 Interrupts</b> .....	<b>1259</b>
<b>23.7 DMA Interface</b> .....	<b>1261</b>
<b>23.8 Module Configuration</b> .....	<b>1262</b>
<b>23.9 Control Registers</b> .....	<b>1264</b>
<b>23.10 Multi-Buffer RAM</b> .....	<b>1332</b>
<b>23.11 Parity Memory</b> .....	<b>1339</b>
<b>23.12 MibSPI Pin Timing Parameters</b> .....	<b>1342</b>

## 23.1 Overview

The MibSPI/MibSPIP is a high-speed synchronous serial input/output port that allows a serial bit stream of programmed length (2 to 16 bits) to be shifted in and out of the device at a programmed bit-transfer rate. Typical applications for the SPI include interfacing to external peripherals, such as I/Os, memories, display drivers, and analog-to-digital converters.

The SPI has the following attributes:

- 16-bit shift register
- Receive buffer register
- 8-bit baud clock generator, supports max up to 20Mhz baud rate
- Serial clock (**SPICLK**) pin
- 4 **SPISOMI/SPISIMO pins for data transfer, with programmable pin direction**
- SPI enable ( $\overline{\text{SPIENA}}$ ) pin
- 4 slave chip select ( $\overline{\text{SPISCS}}[3:0]$ ) 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 Pin  
SOMI - Slave Out Master In Pin  
CS - SPI Chip Select Pin  
ENA - SPI Enable Pin.

This device contains two MibSPIs (MibSPI1 and MibSPI3) and one MibSPIP (MibSPIP5). All three modules support 4 SPICS, 1 SPICLK, 1 SPIENA pins. MibSPI 1 & 3 support 1 SPISOMI/SPISIMO pins; MibSPIP5 supports 4 SPISOMI/SPISIMO pins.

---

### 23.1.1 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

### 23.1.2 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.

#### 23.1.2.1 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).

### 23.1.2.2 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.

---

### 23.1.3 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.

## 23.2 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 ( $\overline{\text{SPISCS}}[3:0]$ ) 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.

### 23.2.1 Pin Configurations

The SPI supports data connections as shown in [Table 23-1](#).

**Table 23-1. 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	<b>SPIENA disabled:</b> GIO	<b>SPIENA enabled:</b> Receives ENA signal from the external slave	<b>SPIENA disabled:</b> GIO	<b>SPIENA enabled:</b> Receives ENA signal from the external master
SPICS[3:0]	<b>SPICS disabled:</b> GIO	<b>SPICS enabled:</b> Selects one or more slave devices	<b>SPICS disabled:</b> GIO	<b>SPICS enabled:</b> Receives the CS signal from the external master

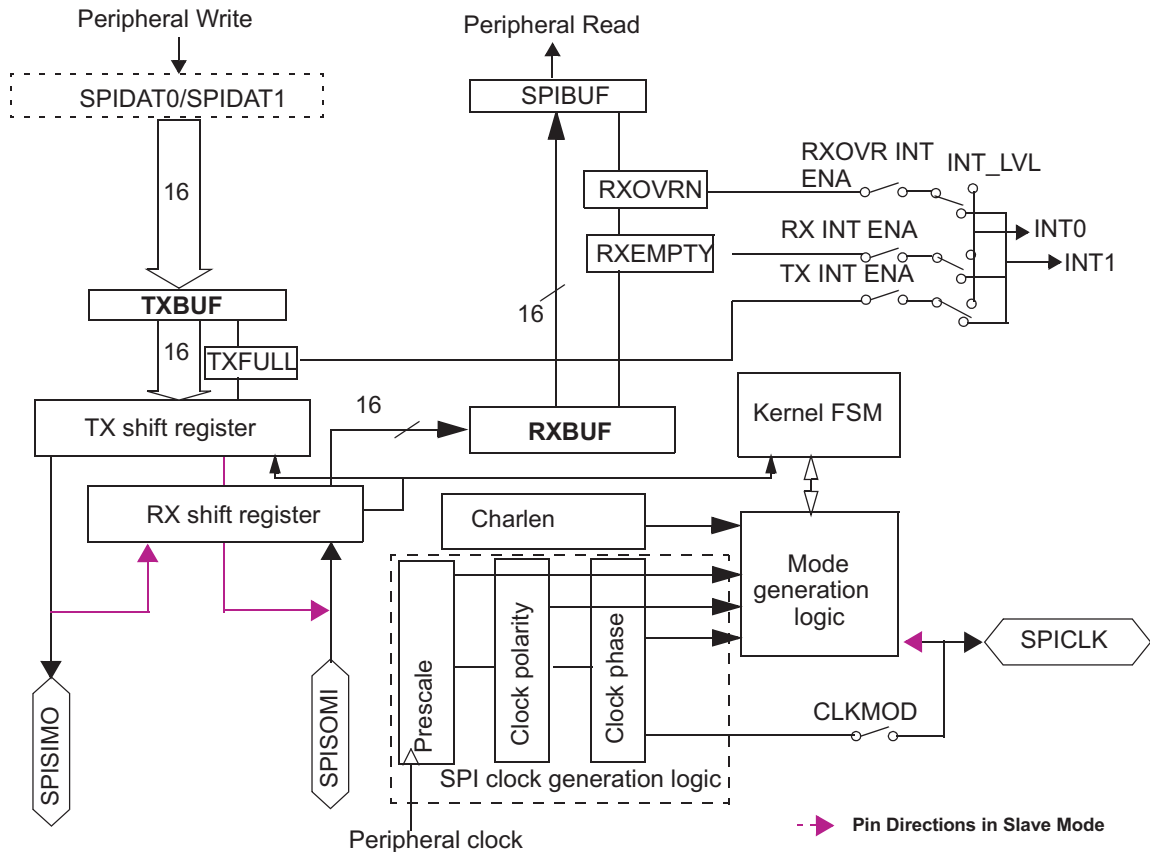
**NOTE:**

1. When the SPICS[3:0] 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.

### 23.2.2 Data Handling

[Figure 23-1](#) 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 23-1. 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.

### 23.2.2.1 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 empty 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.

### 23.2.2.2 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.

### 23.2.2.3 Three-Pin Mode

In master mode configuration (MASTER = 1 and CLKMOD = 1), the SPI provides the serial clock on the SPICLK pin. Data is transmitted on the SPISIMO pin and received on the SPISOMI pin (see Figure 23-2).

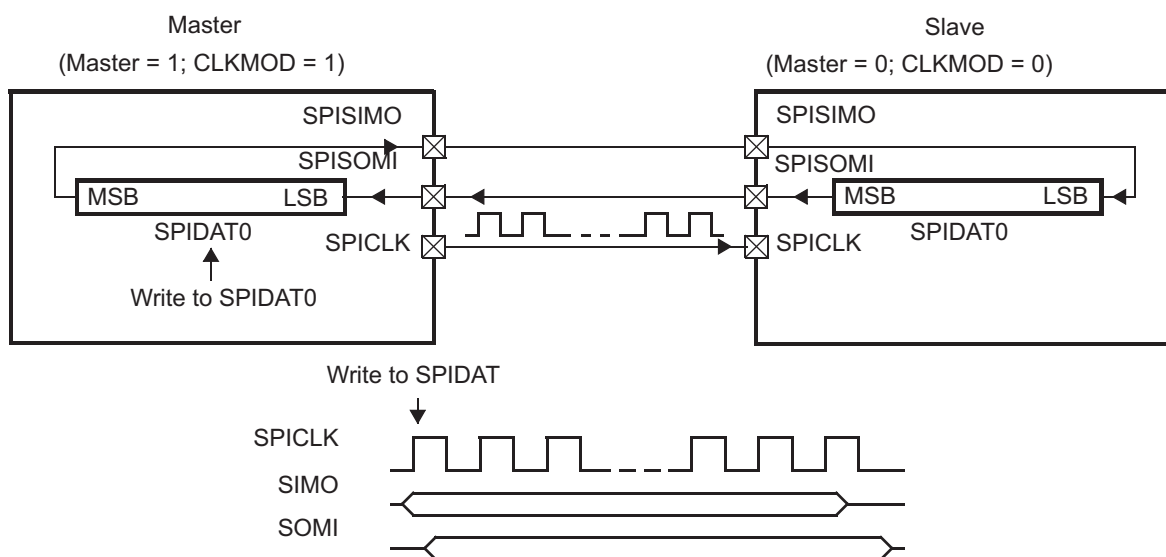
Data written to the shift register (SPIDAT0 / SPIDAT1) initiates data transmission on the SPISIMO pin, MSB first. Simultaneously, received data is shifted through the SPISOMI pin into the LSB of the SPIDAT0 register. When the selected number of bits have been transmitted, the received data in the shift register is transferred to the SPIBUF register for the CPU to read. Data is stored right-justified in SPIBUF.

See Section 23.2.2.1 and Section 23.2.2.2 for details about the data handling for transmit and receive operations.

In slave mode configuration (MASTER = 0 and CLKMOD = 0), data shifts out on the SPISOMI pin and in on the SPISIMO pin. The SPICLK pin is used as the input for the serial shift clock, which is supplied from the external network master. The transfer rate is defined by this clock.

Data written to the SPIDAT0 or SPIDAT1 register is transmitted to the network when the SPICLK signal is received from the network master. To receive data, the SPI waits for the network master to send the SPICLK signal and then shifts data on the SPISIMO pin into the RX shift register. If data is to be transmitted by the slave simultaneously, it must be written to the SPIDAT0 or SPIDAT1 register before the beginning of the SPICLK signal.

**Figure 23-2. SPI Three-Pin Operation**



### 23.2.3 Operation with 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 SPISCS [3:0] pins as chip selects. It is disabled by setting all SPISCS [3:0] pins as GIOs in SPIPC0[3:0]

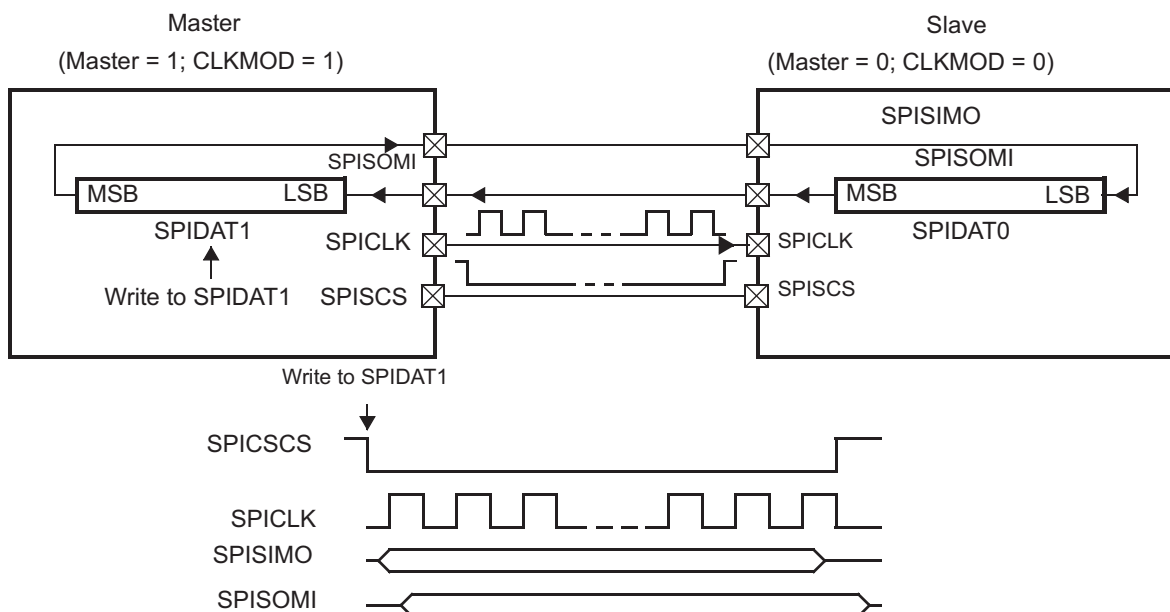
#### Multiple Chip Selects

The SPISCS [3:0] pins that are used must be configured as functional pins in the SPIPC0[3:0] register. The default pattern to be put on the SPISCS [3:0] 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[3:0] output pin. The drive state for SPISCS[3:0] 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  $\overline{\text{SPICSCS}}$  input pin.

**Figure 23-3. Operation with  $\overline{\text{SPICSCS}}$**



### 23.2.4 Operation with $\overline{\text{SPIENA}}$

The  $\overline{\text{SPIENA}}$  operates as a WAIT signal pin. For both the slave and the master, the  $\overline{\text{SPIENA}}$  pin must be configured to be functional ( $\text{SPIPC0}[8] = 1$ ). In this mode, an active low signal from the slave on the  $\overline{\text{SPIENA}}$  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).

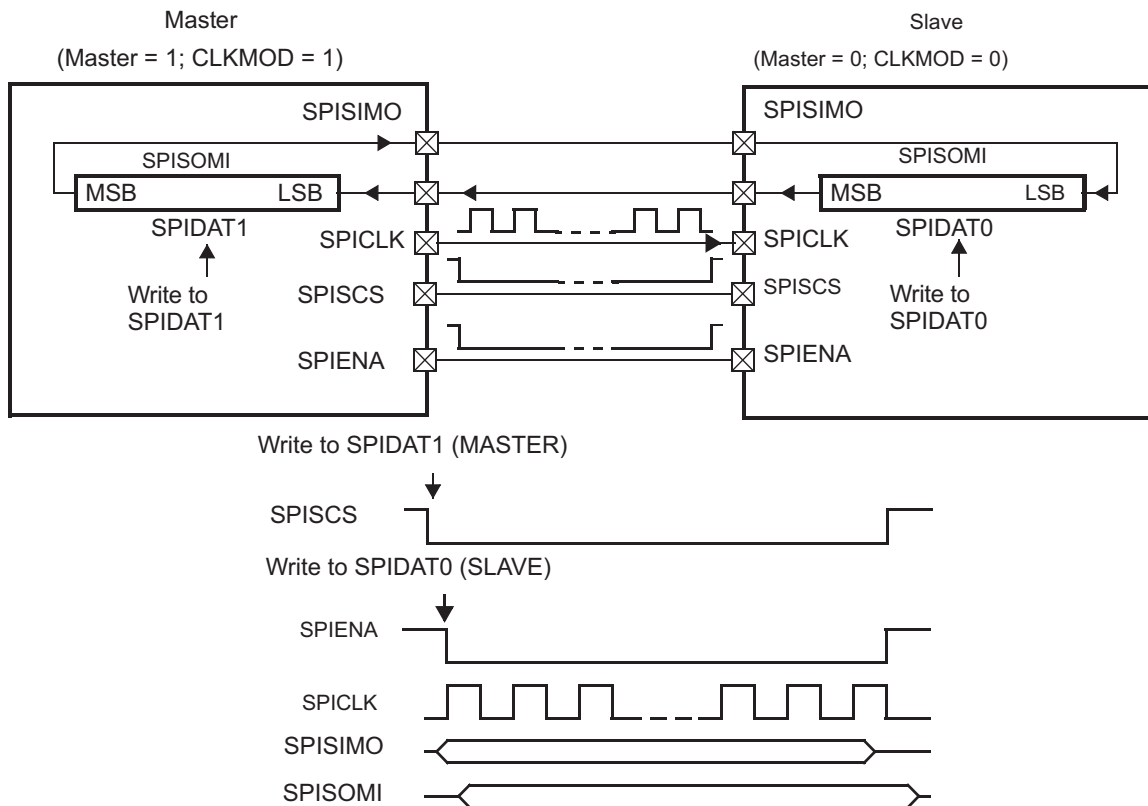
If the  $\overline{\text{SPIENA}}$  pin is in high-z mode ( $\text{ENABLE\_HIGHZ} = 1$ ), the slave will put  $\overline{\text{SPIENA}}$  into the high-impedance once it completes receiving a new character. If the  $\overline{\text{SPIENA}}$  pin is in push-pull mode ( $\text{ENABLE\_HIGHZ} = 0$ ), the slave will drive  $\overline{\text{SPIENA}}$  to 1 once it completes receiving a new character. The slave will drive  $\overline{\text{SPIENA}}$  low again for the next word to transfer, after new data is written to the slave TX shift register.

In master mode ( $\text{CLKMOD} = 1$ ), if the  $\overline{\text{SPIENA}}$  pin is configured as functional, then the pin will act as an input pin. If configured as a slave SPI and as functional, the  $\overline{\text{SPIENA}}$  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  $\text{SPISOMI}$  and  $\overline{\text{SPIENA}}$  (if  $\text{ENABLE\_HIGHZ}$  bit is set to 1) in high-z mode. Once this condition has occurred, if a  $\text{SPICLK}$  edge is detected while the chip select is deasserted, then the SPI stops that transfer and sets an  $\text{DLENERR}$  error flag and generates an interrupt (if enabled).



**Figure 23-5. SPI Five-Pin Option with  $\overline{\text{SPIENA}}$  and  $\overline{\text{SPISCS}}$**



### 23.2.6 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 23-6 shows how a 12-bit word (0xEC9) needs to be written to the transmit buffer to be transmitted correctly.

**Figure 23-6. 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 23-7 shows how a 10-bit word (0x0A2) is stored in the buffer once it is received.

**Figure 23-7. 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

### 23.2.7 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.

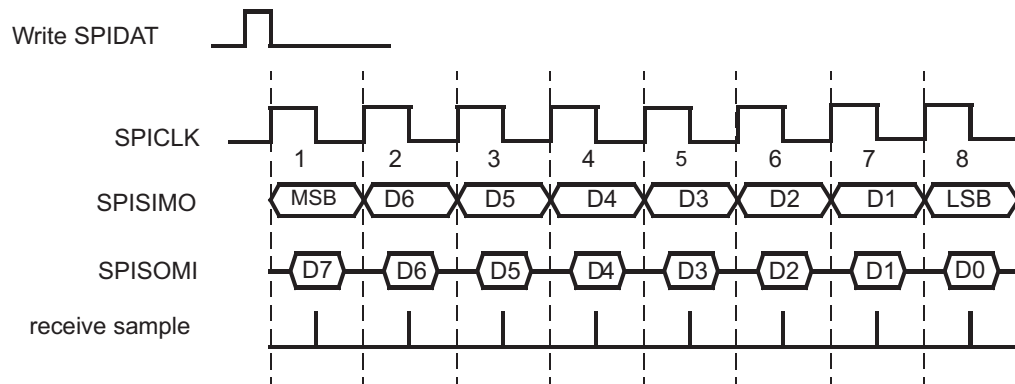
The data input and output edges depend on the values of both **POLARITY** and **PHASE** as shown in Table 23-2.

**Table 23-2. Clocking Modes**

<b>POLARITY</b>	<b>PHASE</b>	<b>ACTION</b>
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.

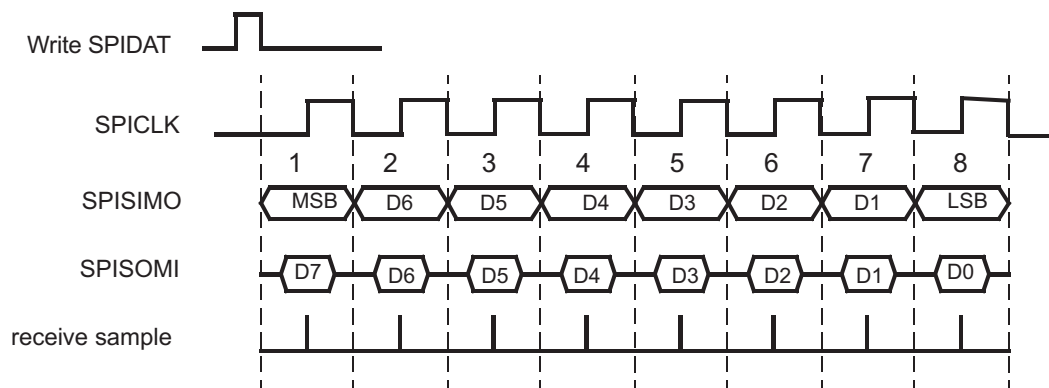
Figure 23-8 to Figure 23-11 illustrate the four possible configurations of **SPICLK** corresponding to each mode. Having four signal options allows the SPI to interface with many different types of serial devices.

**Figure 23-8. Clock Mode with Polarity = 0 and Phase = 0**



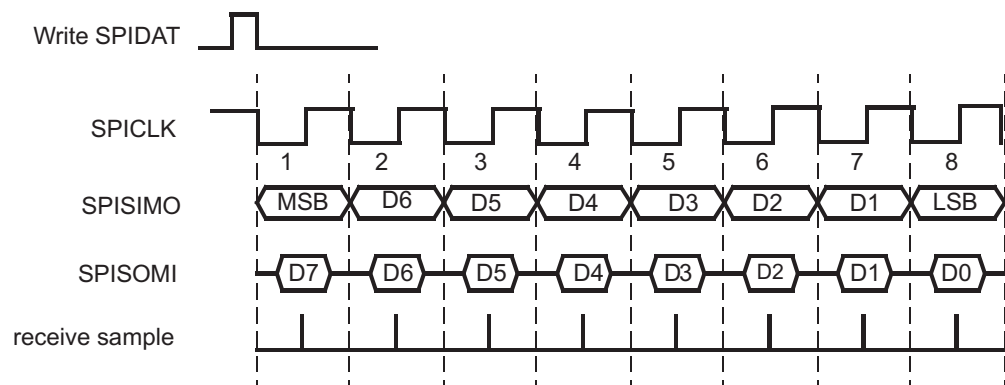
Data is output on the rising edge of SPICLK.  
Input data is latched on the falling edge of SPICLK.

**Figure 23-9. Clock Mode with Polarity = 0 and Phase = 1**



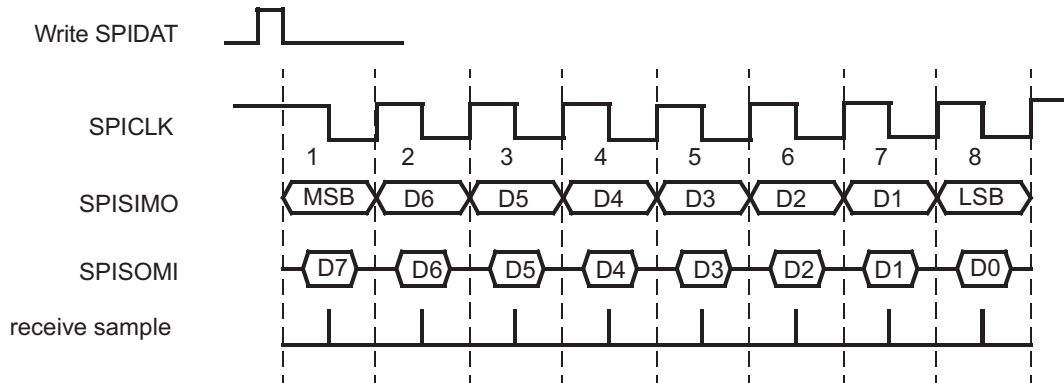
Data is output one-half cycle before the first rising edge of SPICLK and on subsequent falling edges of SPICLK  
Input data is latched on the rising edge of SPICLK

**Figure 23-10. 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 23-11. Clock Mode with Polarity = 1 and Phase = 1**

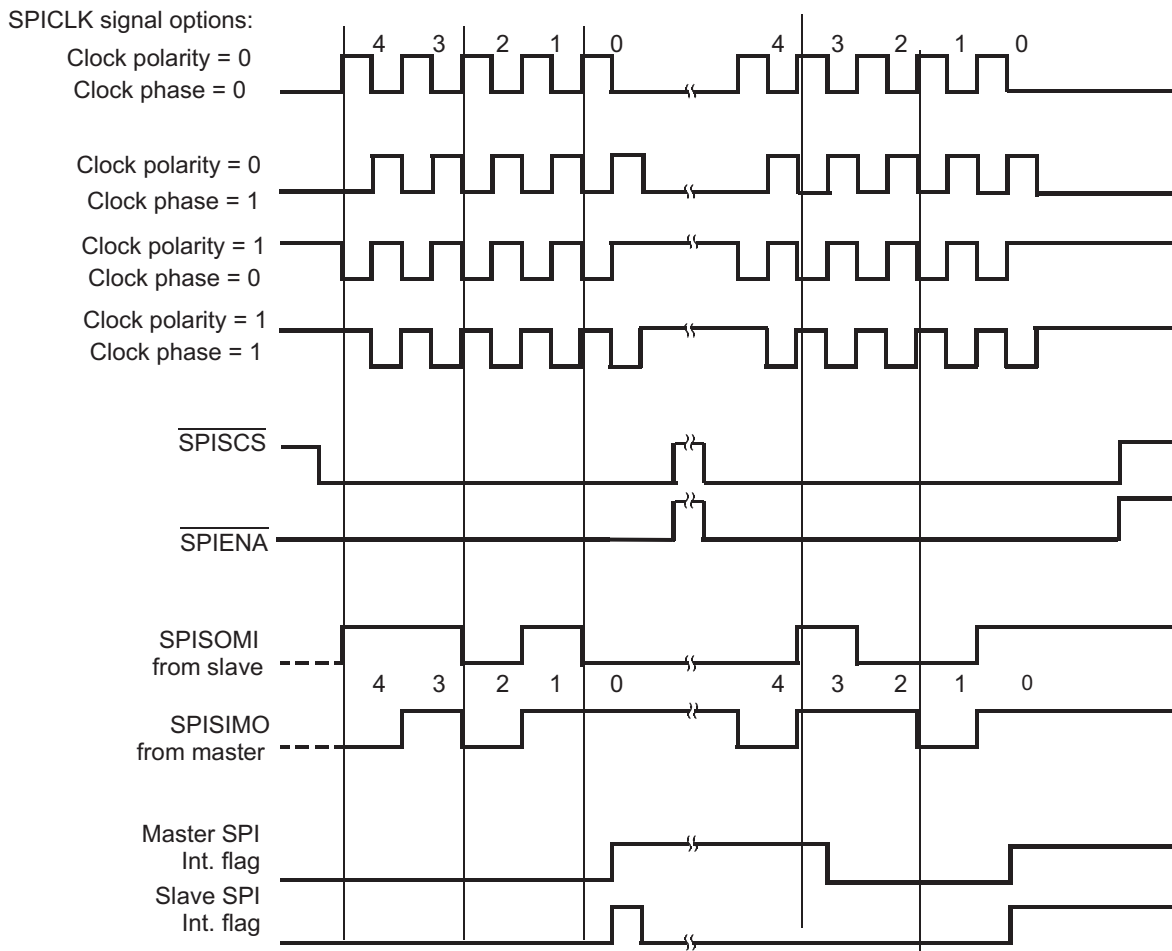


Data is output one-half cycle before the first falling edge of SPICLK and on the subsequent rising edges of SPICLK. Input data is latched on the falling edge of SPICLK.

### 23.2.8 Data Transfer Example

Figure 23-12 illustrates a SPI data transfer between two devices using a character length of five bits.

**Figure 23-12. Five Bits per Character (5-Pin Option)**



### 23.2.9 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  $\overline{\text{SPI}}\text{SCS}$  [4:0] pins. When the transmission finishes the default chip-select value (defined by the CSDEF register) is put on the  $\overline{\text{SPI}}\text{SCS}$  [4:0] 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  $\overline{\text{SPI}}\text{SCS}$  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,  $n$   $\overline{\text{SPI}}\text{SCS}$  pins can be used for encoding a  $n$ -bit address and the remaining pins can be connected to decoded-mode slaves.

### 23.2.10 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.

### 23.2.11 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.

---

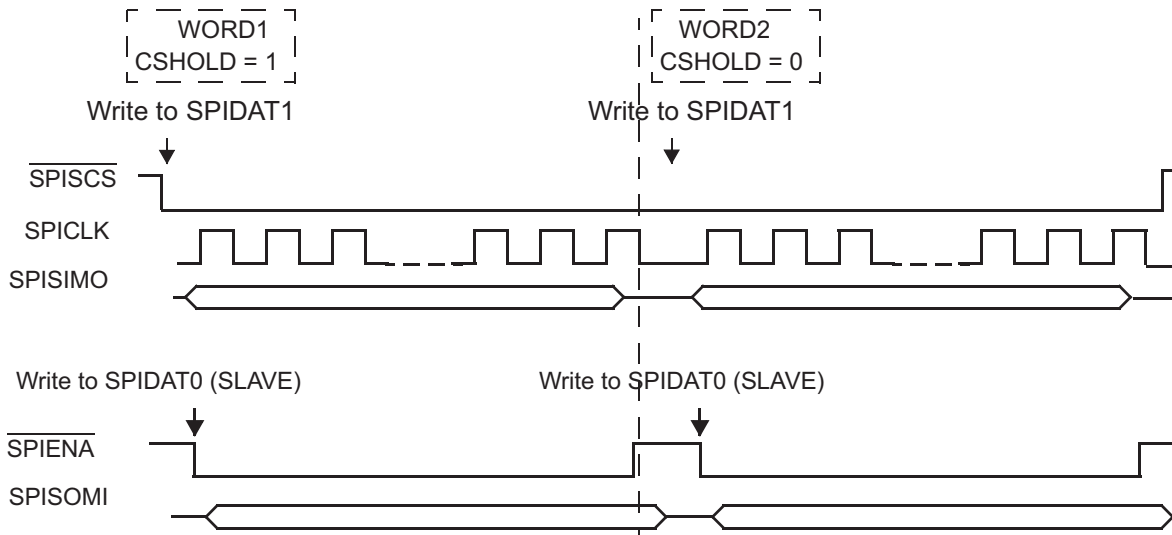
#### 23.2.11.1 The CSHOLD Bit in Master Mode

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 23-13 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 23-13. Typical Diagram when a Buffer in Master is in CSHOLD Mode (SPI-SPI)**


### 23.2.12 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.

---

### 23.2.13 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.

---

### 23.2.14 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  $\overline{\text{SPIEN}}_A$  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  $\overline{\text{SPICLK}}$  line).

---

**NOTE:** In a master mode SPI, the data length error will be generated only if the  $\overline{\text{SPIEN}}_A$  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  $\overline{\text{SPICLK}}$  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.

---

### 23.2.15 Parallel Mode (Multiple SIMO/SOMI Support, not available on all devices)

In order to increase throughput, the parallel mode of the SPI enables the module to send data over more than one data line (Parallel 2, 4 or 8). When parallel mode is used, the data length must be set as 16 bits. Only module MIBSPIP5 supports Parallel Mode.

This feature increases throughput by 2 for 2 pins, by 4 for 4 pins, or by 8 for 8 pins.

Parallel mode supports the following features:

- Scalable data lines (1, 2, 4, 8) per direction. (SOMI and SIMO lines)
- All clock schemes are supported (clock phase and polarity)
- Parity is supported. The parity bit will be transmitted on bit0 of the SIMO/SOMI lines. The receive parity is expected on bit0 of the SOMI/SIMO pins.

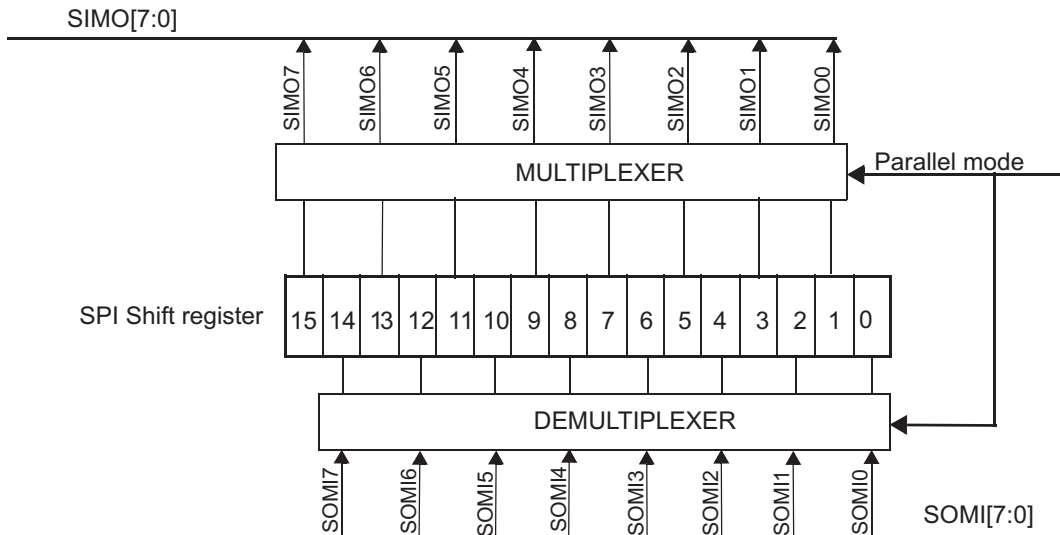
Parallel mode can be programmed using the  $\text{PMODEx}[1:0]$  bits of  $\text{SPIPMCTRL}$  register. See [Section 23.9.24](#) for details about this register.

After reset the parallel mode selection bits are cleared (single SIMO/SOMI lines).

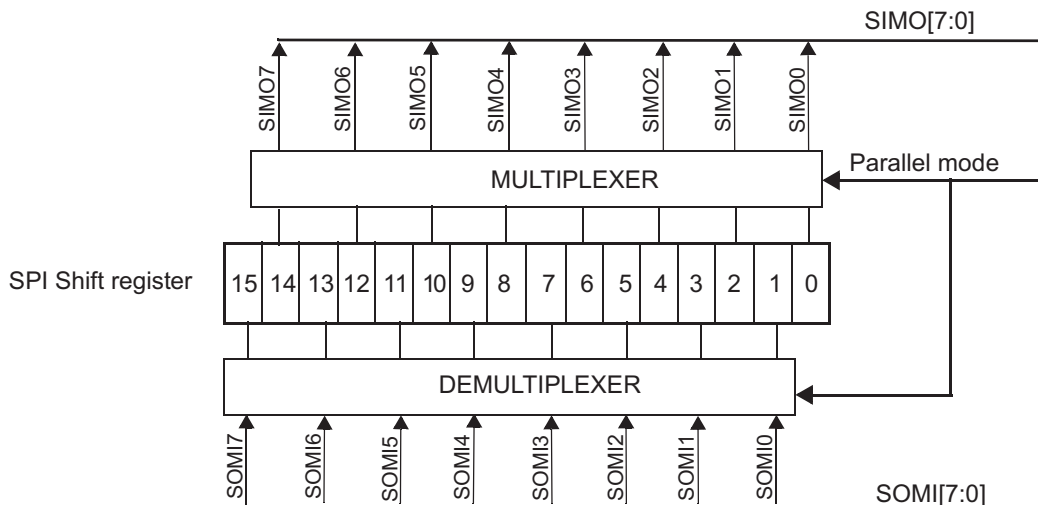
#### 23.2.15.1 Parallel Mode Block Diagram

[Figure 23-14](#) and [Figure 23-15](#) show the parallel connections to the SPI shift register.

**Figure 23-14. Block Diagram Shift Register, MSB First**



**Figure 23-15. Block Diagram Shift Register, LSB First**



**23.2.15.2 Parallel Mode Pin Mapping, MSB First**

Table 23-3 and Table 23-4 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 23-3. Pin Mapping for SIMO Pin with MSB First**

Parallel Mode	Shift Register Bit	SIMO[7:0]
1	15	0
2	15	1
	7	0

**Table 23-3. Pin Mapping for SIMO Pin with MSB First (continued)**

Parallel Mode	Shift Register Bit	SIMO[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 23-4. 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

### 23.2.15.3 Parallel Mode Pin Mapping, MSB-First, LSB-First

Table 23-5–Table 23-6 describe the SIMO and SOMI pin mapping when SPI is used in parallel mode (1, 2, 4, 8) pin mode, LSB first.

**Table 23-5. 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

**Table 23-5. Pin Mapping for SIMO Pin with LSB First (continued)**

Parallel Mode	Shift Register Bit	SIMO[7:0]
8	14	7
	12	6
	10	5
	8	4
	6	3
	4	2
	2	1
	0	0

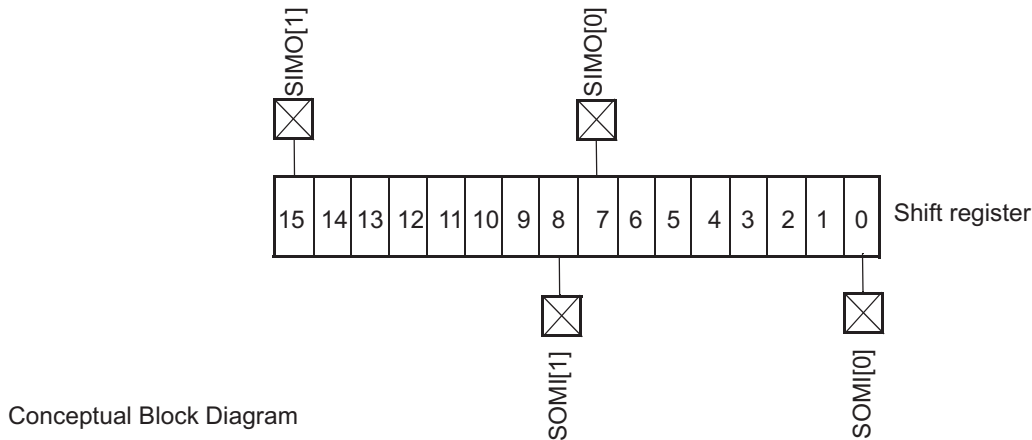
**Table 23-6. Pin Mapping for SOMI Pin with LSB First**

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

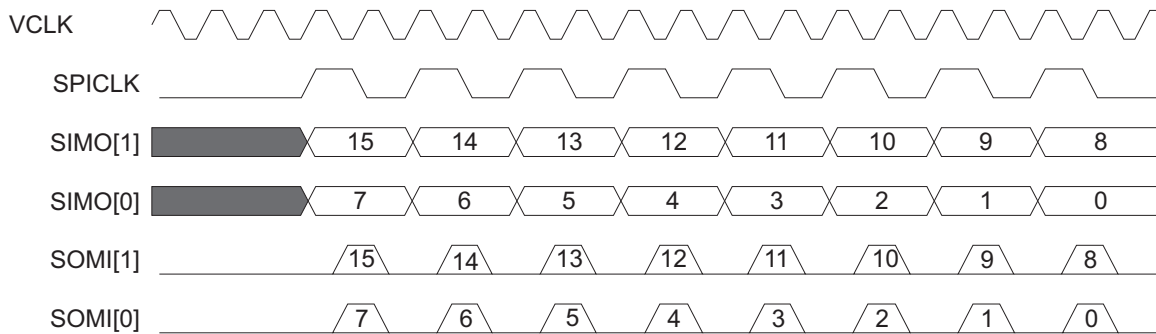
**23.2.15.4 2-Data Line Mode (MSB First, Phase 0, Polarity 0)**

In 2-data line mode (master mode) the shift register bits 15 and 7 will be connected to the pins SIMO[1] and SIMO[0], and the shift register bits 8 and 0 will be connected to the pins SOMI[1] and SOMI[0] or vice versa in slave mode. After writing to the SPIDAT0/SPIDAT1 register, the bits 15 and 7 will be output on SIMO[1] and SIMO[0] on the rising edge of 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 23-16](#) shows the clock /data diagram of the 2-data line mode. [Figure 23-17](#) shows the timing of a two-pin parallel transfer.

**Figure 23-16. 2-data Line Mode (Phase 0, Polarity 0)**



**Figure 23-17. Two-Pin Parallel Mode Timing Diagram (Phase 0, Polarity 0)**

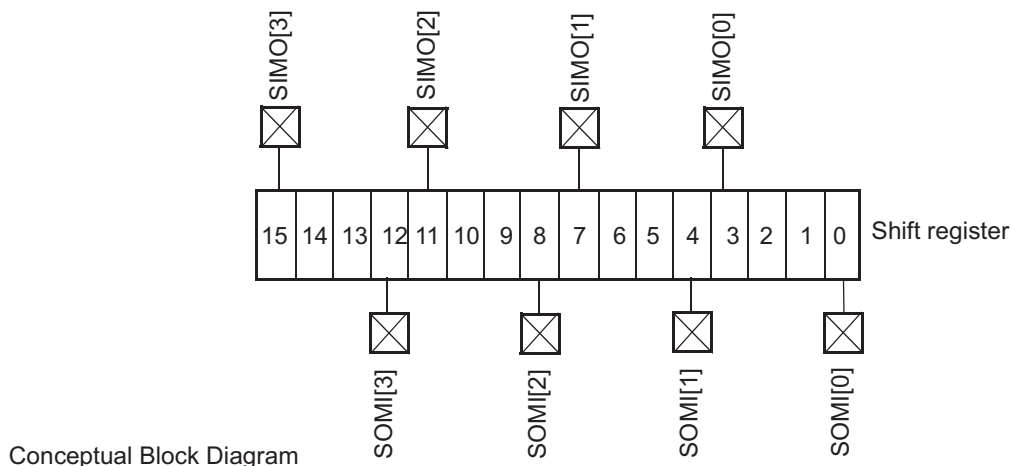


**23.2.15.5 4-Data Line Mode (MSB First, Phase 0, Polarity 0)**

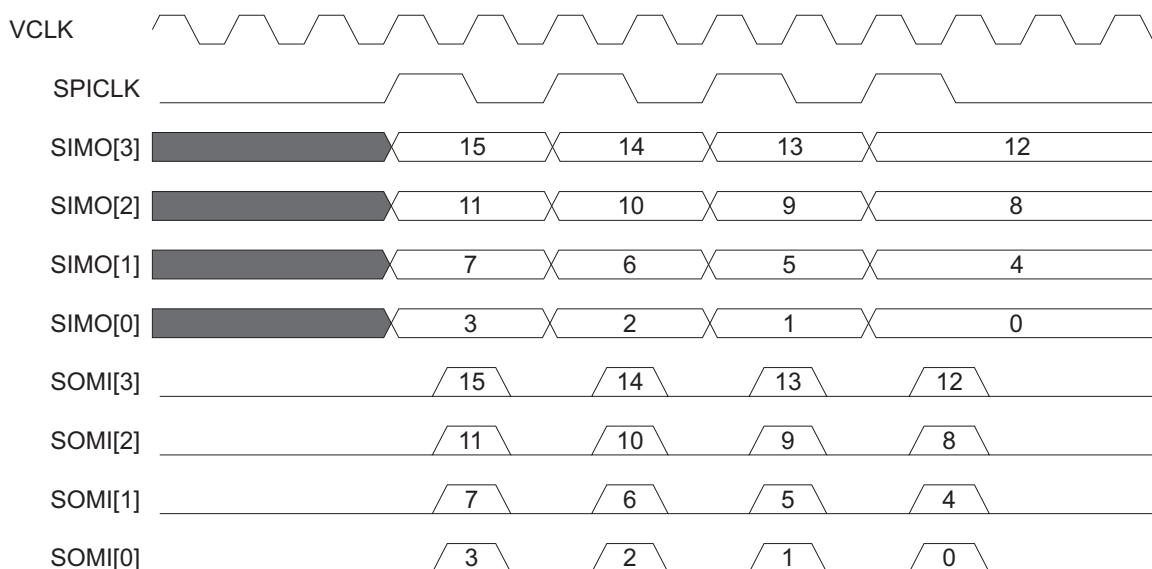
In 4-data line mode (master mode) the shift register bits 15, 11, 7, and 3 will be connected to the pins SIMO[3], SIMO[2], SIMO[1], and SIMO[0], and the shift register bits 12, 8, 4, and 0 will be connected to the pins SOMI[3], SOMI[2], SOMI[1], and SOMI[0] (or vice versa in slave mode). After writing to SPIDAT1/SPIDAT0, the bits 15, 11, 7, and 3 will be output on SIMO[3], SIMO[2], SIMO[1], and SIMO[0] on the rising edge of SPICLK. With the falling clock edge of the SPICLK, the received data on SOMI[3], SOMI[2], SOMI[1] and SOMI[0] will be latched to shift register bits 12, 8, 4, and 0. The subsequent rising edge of SPICLK will shift data in the shift register by 1 bit to the left ( SIMO[3] will shift the data out from bit 15 to 12, SIMO[2] will shift the data out from bit 11 to 8, SIMO[1] will shift the data out from bit 7 to 4, SIMO[0] will shift the data out from bit 3 to 0). After four SPICLK cycles, when the full data word is transferred, the shift register (16 bits) is copied to the receive buffer, and the RXINT flag will be set.

Figure 23-18 shows the clock/data diagram of the four-data line mode. Figure 23-19, shows the timing diagram for four-data line mode.

**Figure 23-18. 4-Data Line Mode (Phase 0, Polarity 0)**



**Figure 23-19. 4 Pins Parallel Mode Timing Diagram (Phase 0, Polarity 0)**



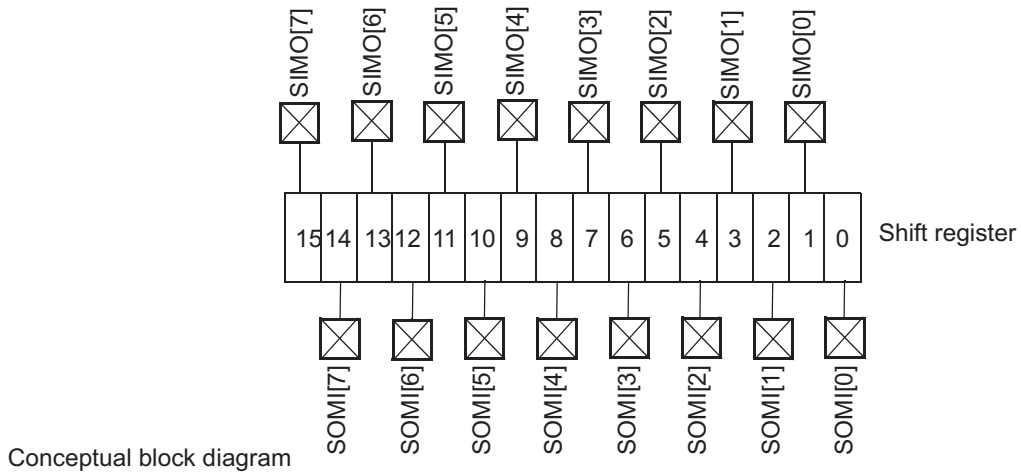
**23.2.15.6 8-Data Line Mode (MSB First, Phase 0, Polarity 0)**

In 8-data line mode (master mode) the shift register bits 15, 13, 11, 9, 7, 5 and 3 will be connected to the pins SIMO[7], SIMO[6], SIMO[5], SIMO[4], SIMO[3], SIMO[2], SIMO[1], and SIMO[0], and the shift-register bits 14, 12, 10, 8, 6, 4, and 0 will be connected to the pins SOMI[7], SOMI[6], SOMI[5], SOMI[4], SOMI[3], SOMI[2], SOMI[1], and SOMI[0] (or vice versa in slave mode).

After writing to SPIDAT0/SPIDAT1, the bits 15, 13, 11, 9, 7, 5, 3, and 1 will be output on SIMO[7], SIMO[6], SIMO[5], SIMO[4], SIMO[3], SIMO[2], SIMO[1], and SIMO[0], on the rising edge of SPICLK. On the falling clock edge of the SPICLK, the received data on SOMI[8], SOMI[7], SOMI[6], SOMI[5], SOMI[4], SOMI[3], SOMI[2], SOMI[1], and SOMI[0] will be latched to the shift register bits 14, 12, 10, 8, 6, 4, 2, and 0.

The subsequent rising edge of SPICLK will shift the data in the shift register by 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 23-20](#) shows the clock/data diagram of the 8-data line mode. [Figure 23-21](#) shows the pin timings for 8-data line mode.

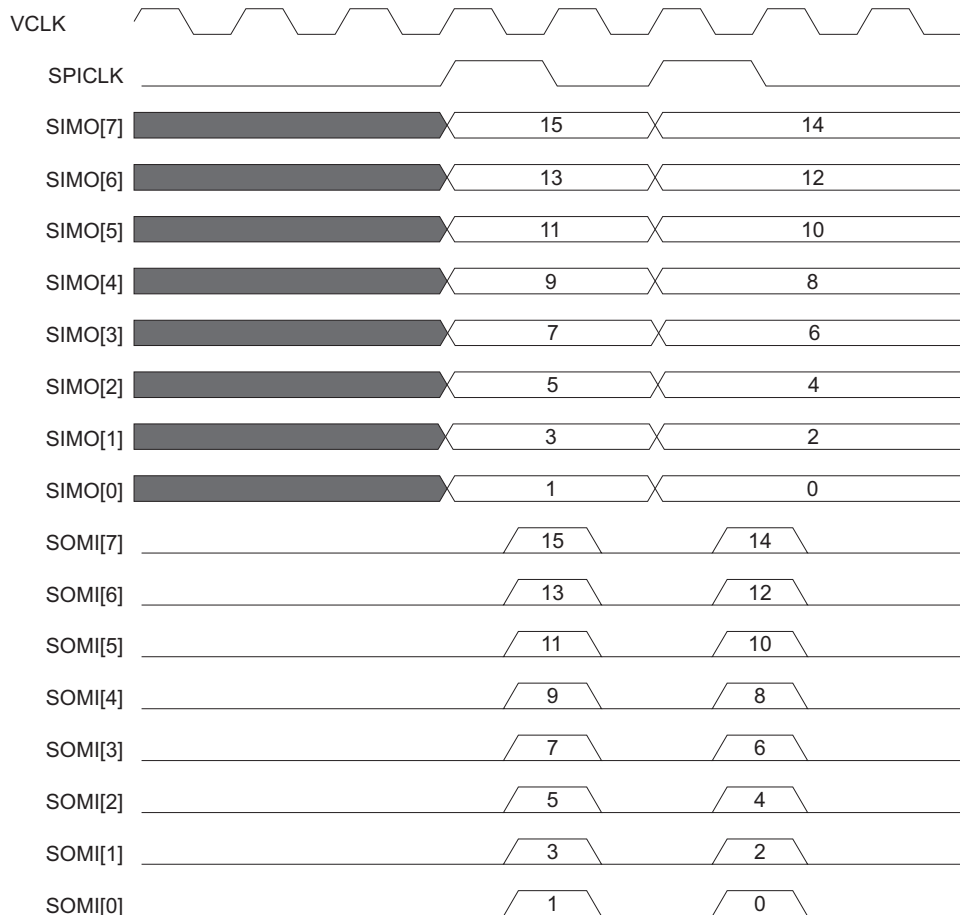
**Figure 23-20. Eight-data Line Mode (Phase 0, Polarity 0)**



**NOTE: Parity Support**

Using the parity support in parallel mode may seriously affect throughput. For an eight-line mode to transfer 16 bits of data, only two SPICLK pulses are enough. If parity is enabled, one extra SPICLK pulse will be used to transfer and receive the parity bit. Parity will be transmitted and received on the 0th line regardless of 1/2/4/8-line modes. During the parity bit transfer, other data bits are not valid.



**Figure 23-21. 8 Pins Parallel Mode Timing Diagram (Phase 0, Polarity 0)**

**NOTE:** Modulo Count Parallel Mode is not supported in this device.

### 23.2.16 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.

## 23.3 Test Features

### 23.3.1 Internal Loop-Back Test Mode (Master Only)

The internal loop-back self-test mode can be utilized to test the SPI transmit and receive paths, including the shift registers, the SPI buffer registers, and the parity generator. In this mode the transmit signal is internally feedback to the receiver, whereas the SIMO, SOMI, and CLK pin are disconnected, 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.

---

### 23.3.2 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 23-22](#) for a diagram of the types of feedback available. The IOLPBKTSTCR register defines all of the available control fields.

In loopback mode, it is also possible to induce various error conditions. See [Section 23.9.42](#) 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.

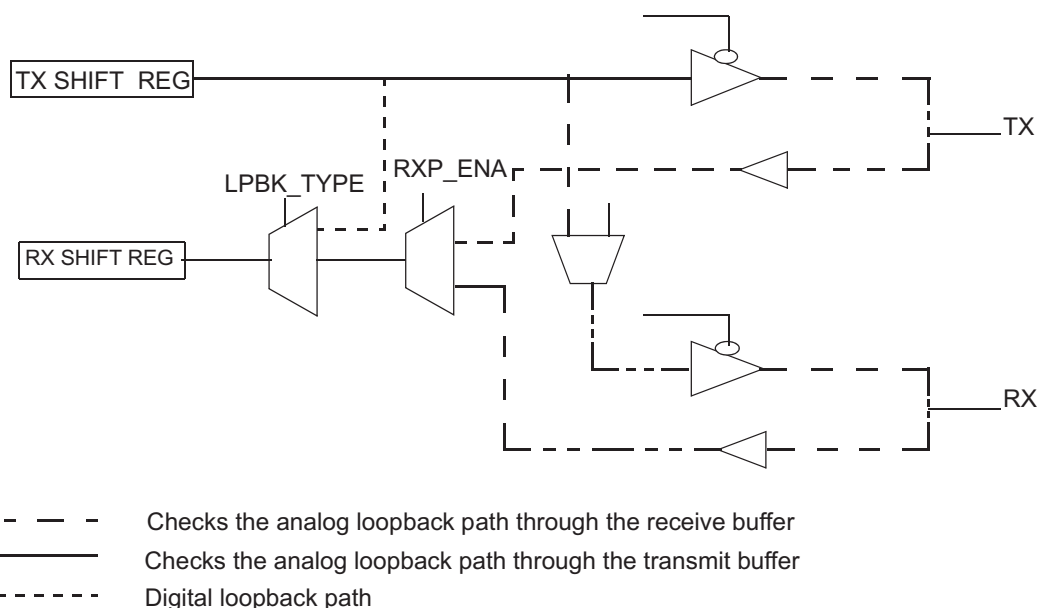
In Input/Output loopback test modes, if the module is in master mode, the nENA signal is also generated by internal logic so that an external interface is not required.

---

**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 23-22. 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.

### 23.3.2.1 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  $\overline{\text{SPISCS}}[3:0]$  to 0x0. The actual number of chip selects can be programmed to have any or all of the  $\overline{\text{SPISCS}}$  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.

## 23.4 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)

## 23.5 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.

---

## 23.6 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.

---

### 23.6.1 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:

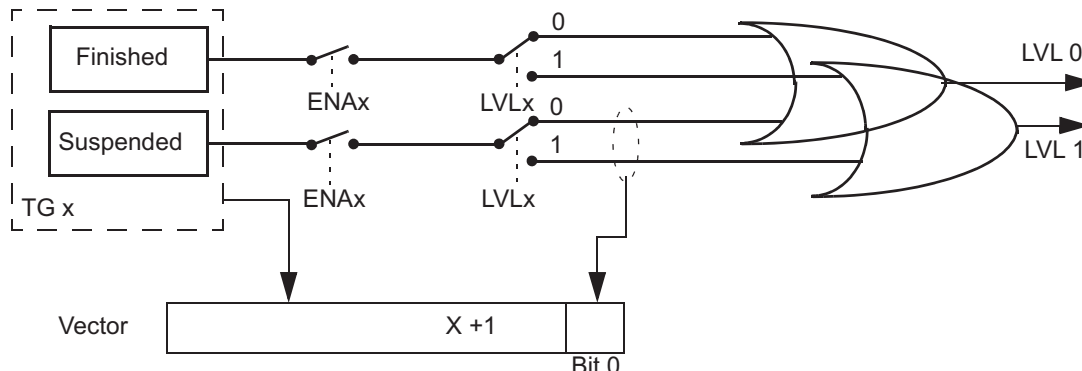
- Transmission error interrupt
- Receive overrun interrupt
- TG suspended interrupt
- TG completed interrupt

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 23-23](#) illustrates the TG interrupts.

Figure 23-23. TG Interrupt Structure

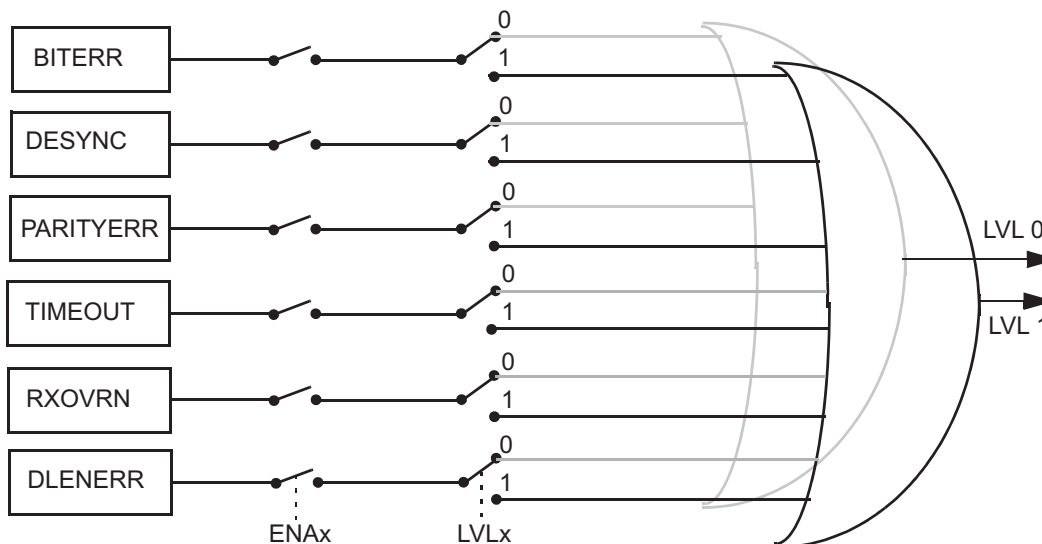


During transmission, if one of the following errors occurs, BITERR, DESYNC, PARITYERR, TIMEOUT, DLENERR, the corresponding flag in the SPIFLG register is set. If the enable bit is set, then an interrupt is generated. The level of the interrupts could be generated according to the bit field in SPILVL register.

The RXOVRN interrupt is generated when a buffer in the RXRAM is overwritten by a new received word. While writing newly received data to a RXRAM location, if the RXEMPTY bit of the corresponding location is 0, then the RXOVR bit will be set to 1 during the write operation, so that the buffer starts to indicate an overrun. This RXOVR flag is also reflected in SPIFLG register as RXOVRNINTFLG and the corresponding vector number is updated in TGINTVECT0/TGINTVECT1 register. If an overrun interrupt is enabled, then an interrupt will be generated indicating an overrun condition.

The error interrupts are enabled and prioritized independently from each other, but the vector generated by the SPI will be the same if multiple errors are enabled on the same level.

Figure 23-24. SPIFLG Interrupt Structure



Since the priority of an error interrupt is lower than a completion/suspend interrupt for a TG, the interrupts can be split into two levels. By programming all the error interrupts into Level 0 and TG-complete / TG-suspend interrupts into Level 1, it is possible to get a clear indication of the source of error interrupts. However, when a vector register shows an error interrupt, the actual buffer for which the error has occurred is not readily identifiable. Since each buffer in the multi-buffer RAM is stored along with its individual status flags, each buffer should be read until a buffer with any error flag set is found.

A separate interrupt line is provided to indicate the uncorrectable error condition in the MibSPI. This line is available (and valid) only in the multi-buffer mode of the MibSPI module and if the parity error detection feature for multi-buffer RAM is enabled.

## 23.7 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.

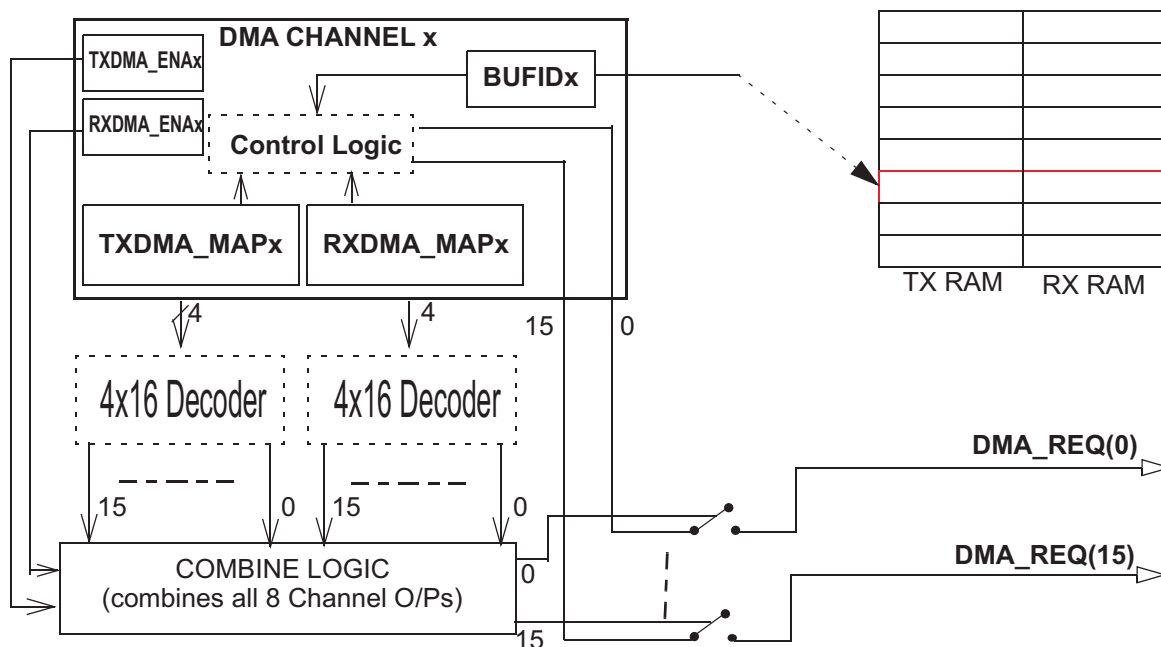
### 23.7.1 DMA in Multi-Buffer Mode

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 23-25. DMA Channel and Request Line (Logical) Structure in Multibuffer Mode





## 23.8 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.

### 23.8.1 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.

### 23.8.2 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 occurrence of the correct trigger event the Transfer group will be triggered and data gets transmitted and received one after the other with out any CPU intervention.
- User can poll Transfer-group interrupt flag or wait for a transfer-completed interrupt to read and write new data to the buffers.

## 23.9 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 23-7. SPI Registers**

Offset	Acronym	Register Description	Section
00	SPIGCR0	SPI Global Control Register 0	<a href="#">Section 23.9.1</a>
04h	SPIGCR1	SPI Global Control Register 1	<a href="#">Section 23.9.2</a>
08h	SPIINT0	SPI Interrupt Register	<a href="#">Section 23.9.3</a>
0Ch	SPIILVL	SPI Interrupt Level Register	<a href="#">Section 23.9.4</a>
10h	SPIFLG	SPI Flag Register	<a href="#">Section 23.9.5</a>
14h	SPIPC0	SPI Pin Control Register 0	<a href="#">Section 23.9.6</a>
18h	SPIPC1	SPI Pin Control Register 1	<a href="#">Section 23.9.7</a>
1Ch	SPIPC2	SPI Pin Control Register 2	<a href="#">Section 23.9.8</a>
20h	SPIPC3	SPI Pin Control Register 3	<a href="#">Section 23.9.9</a>
24h	SPIPC4	SPI Pin Control Register 4	<a href="#">Section 23.9.10</a>
28h	SPIPC5	SPI Pin Control Register 5	<a href="#">Section 23.9.11</a>
2Ch	SPIPC6	SPI Pin Control Register 6	<a href="#">Section 23.9.12</a>
30h	SPIPC7	SPI Pin Control Register 7	<a href="#">Section 23.9.13</a>
34h	SPIPC8	SPI Pin Control Register 8	<a href="#">Section 23.9.14</a>
38h	SPIDAT0	SPI Transmit Data Register 0	<a href="#">Section 23.9.15</a>
3Ch	SPIDAT1	SPI Transmit Data Register 1	<a href="#">Section 23.9.16</a>
40h	SPIBUF	SPI Receive Buffer Register	<a href="#">Section 23.9.17</a>
44h	SPIEMU	SPI Emulation Register	<a href="#">Section 23.9.18</a>
48h	SPIDELAY	SPI Delay Register	<a href="#">Section 23.9.19</a>
4Ch	SPIDEF	SPI Default Chip Select Register	<a href="#">Section 23.9.20</a>
50h-5Ch	SPIFMT	SPI Data Format Registers	<a href="#">Section 23.9.21</a>
60h	INTVECT0	Interrupt Vector 0	<a href="#">Section 23.9.22</a>
64h	INTVECT1	Interrupt Vector 1	<a href="#">Section 23.9.23</a>
6Ch	SPIPMCTRL	Parallel/Module Mode Control Register	<a href="#">Section 23.9.24</a>
70h	MIBSPIE	Multi-buffer Mode Enable Register	<a href="#">Section 23.9.25</a>
74h	TGITENST	TG Interrupt Enable Set Register	<a href="#">Section 23.9.26</a>
78h	TGITENCR	TG Interrupt Enable Clear Register	<a href="#">Section 23.9.27</a>
7Ch	TGITLVST	Transfer Group Interrupt Level Set Register	<a href="#">Section 23.9.28</a>
80h	TGITLVCR	Transfer Group Interrupt Level Clear Register	<a href="#">Section 23.9.29</a>
84h	TGINTFLG	Transfer Group Interrupt Flag Register	<a href="#">Section 23.9.30</a>
90h	TICKCNT	Tick Count Register	<a href="#">Section 23.9.31</a>
94h	LTGPEND	Last TG End Pointer	<a href="#">Section 23.9.32</a>
98h-D4h	TGxCTRL	TGx Control Registers	<a href="#">Section 23.9.33</a>
D8h-F4h	DMAxCTRL	DMA Channel Control Register	<a href="#">Section 23.9.34</a>
F8h-114h	ICOUNT	DMAxCOUNT Register	<a href="#">Section 23.9.35</a>
118h	DMACNTLEN	DMA Large Count	<a href="#">Section 23.9.36</a>
120h	UERRCTRL	Multi-buffer RAM Uncorrectable Parity Error Control Register	<a href="#">Section 23.9.37</a>
124h	UERRSTAT	Multi-buffer RAM Uncorrectable Parity Error Status Register	<a href="#">Section 23.9.38</a>
128h	UERRADDR1	RXRAM Uncorrectable Parity Error Address Register	<a href="#">Section 23.9.39</a>
12Ch	UERRADR0	TXRAM Uncorrectable Parity Error Address Register	<a href="#">Section 23.9.40</a>
130h	RXOVRN_BUF_ADDR	RXRAM Overrun Buffer Address Register	<a href="#">Section 23.9.41</a>
134h	IOLPBKTSTCR	I/O Loopback Test Control Register	<a href="#">Section 23.9.42</a>

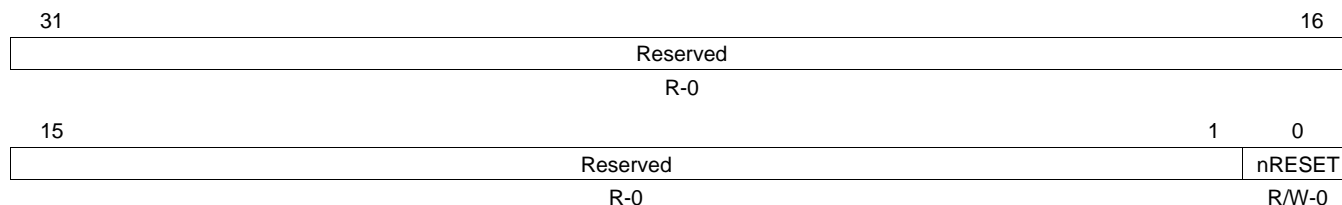
---

**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.

---

### 23.9.1 SPI Global Control Register 0 (SPIGCR0)

Figure 23-26. SPI Global Control Register 0 (SPIGCR0) [offset = 00]



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

Table 23-8. SPI Global Control Register 0 (SPIGCR0) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Read returns 0. Writes have no effect.
0	nRESET		This is the local reset control for the module. This bit needs to be set to '1' before any operation on SPI / MibSPI can be done. Only after setting this bit to '1', the Auto Initialization of Multibuffer RAM starts. Clearing this bit to '0' will result in all of the control and status register values to return to their default values..
		0	SPI is in the reset state.
		1	SPI is out of the reset state.

**23.9.2 SPI Global Control Register 1 (SPIGCR1)**
**Figure 23-27. SPI Global Control Register 1 (SPIGCR1) [offset = 04h]**

31	25	24	23	17	16
Reserved		SPIEN			
R-0		R/W-0	R/W-0		
15	9	8	7	2	1 0
Reserved		POWER DOWN			CLKM OD MAST ER
R-0		R/W-0	R/W-0		R/W-0

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

**Table 23-9. SPI Global Control Register 1 (SPIGCR1) Field Descriptions**

Bit	Field	Value	Description
31-25	Reserved	0	Read returns 0. Writes have no effect.
24	SPIEN	0 1	<p>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:</p> <ul style="list-style-type: none"> <li>• Both TX and RX shift registers</li> <li>• The TXDATA fields of Error: Reference source not found and Error: Reference source not found registers</li> <li>• All the fields of the SPI Flag Register (SPIFLG) register</li> <li>• Contents of SPIBUF and the internal RXBUF registers</li> </ul> <p>0 The SPI is not activated for transfers. 1 Activates SPI</p>
23-17	Reserved	0	Reads returns 0. Writes have no effect.
16	LOOPBACK	0 1	<p>Internal loop-back test mode. The internal self-test option can be enabled by setting this bit. If the SPISIMO and SPISOMI pins are configured with SPI functionality, then the SPISIMO[7:0] pins are internally connected to the SPISOMI[7:0] pins (transmit data is looped back as receive data). GIO mode for these pins is not supported in loopback mode. Externally, during loop-back operation, the SPICLK pin outputs an inactive value and SPISOMI[7:0] remains in the high-impedance state. If the SPI is initialized in slave mode or a data transfer is ongoing, errors may result.</p> <p><b>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.</b></p> <p>0 Internal loop-back test mode disabled. 1 Internal loop-back test mode enabled.</p>
15-9	Reserved	0	Reads returns 0. Writes have no effect.
8	POWER DOWN	0 1	<p>When active, the SPI state machine enters a power-down state.</p> <p>0 The SPI is in active mode. 1 The SPI is in power-down mode.</p>
7-2	Reserved	0	Reads returns 0. Writes have no effect.
1	CLKMOD	0 1	<p>Clock mode. This bit selects either an internal or external clock source. This bit also determines the I/O direction of the <math>\overline{\text{SPIENA}}</math> and <math>\overline{\text{SPISCS}}[3:0]</math> pins in functional mode.</p> <p>0 Clock is external.</p> <ul style="list-style-type: none"> <li>• <math>\overline{\text{SPIENA}}</math> is an output.</li> <li>• <math>\overline{\text{SPISCS}}[3:0]</math> are inputs.</li> </ul> <p>1 Clock is internally-generated.</p> <ul style="list-style-type: none"> <li>• <math>\overline{\text{SPIENA}}</math> is an output.</li> <li>• <math>\overline{\text{SPISCS}}[3:0]</math> are outputs.</li> </ul>

**Table 23-9. SPI Global Control Register 1 (SPIGCR1) Field Descriptions (continued)**

Bit	Field	Value	Description
0	MASTER		<p>SPI SIMO/SPI SOMI pin direction determination. Sets the direction of the SPI SIMO and SPI SOMI pins.</p> <p><b>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.</b></p> <p><b>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.</b></p>
		0	SPI SIMO[7:0] pins is anare inputs, SPI SOMI[7:0] pins is anare outputs
		1	SPI SOMI[7:0] pins is anare inputs, SPI SIMO[7:0] pins is anare outputs

**23.9.3 SPI Interrupt Register (SPIINT0)**
**Figure 23-28. SPI Interrupt Register (SPIINT0) [offset = 08h]**

31	25				24	23					17	16		
Reserved					ENAB LE HIGHZ	Reserved					DMAR EQEN			
R-0					R/W-0		R-0					R/W-0		
15	10				9	8	7	6	5	4	3	2	1	0
Reserved					TXINT ENA	RXINT ENA	Reserv ed	RXOV RNINT ENA	Reserv ed	BITER RENA	DESY NCEN A	PARE RREN A	TIMEO UT ENA	DLEN ERR ENA
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

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

**Table 23-10. SPI Interrupt Register (SPIINT0) Field Descriptions**

Bit	Field	Value	Description
31-25	Reserved	0	Read returns 0. Writes have no effect.
24	ENABLEHIGHZ	0	$\overline{\text{SPIENA}}$ 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.
		1	$\overline{\text{SPIENA}}$ pin is pulled high when not active. $\overline{\text{SPIENA}}$ pin remains high-impedance when not active.
23-17	Reserved	0	Read returns 0. Writes have no effect.
16	DMAREQEN	0	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. DMA is not used.
		1	DMA requests will be generated. <b>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.</b> <b>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.</b>
15-10	Reserved	0	Read returns 0. Writes have no effect.
9	TXINTENA	0	Causes an interrupt to be generated every time data is written to the shift register, so that the next word can be written to TXBUF. Setting this bit will generate an interrupt if the TXINTFLG bit (SPI Flag Register (SPIFLG)[9]) is set to 1. No interrupt will be generated upon TXINTFLG being set to 1.
		1	An interrupt will be generated upon TXINTFLG being set to 1. The transmitter empty interrupt is valid in compatibility mode of SPI only. In multi-buffered mode, this interrupts will not be generated, even if it is enabled. <b>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.</b>
8	RXINTENA	0	Causes an interrupt to be generated when the RXINTFLAG bit (SPI Flag Register (SPIFLG)[8]) is set by hardware. Interrupt will not be generated.
		1	Interrupt will be generated. The receiver full interrupt is valid in compatibility mode of SPI only. In multi-buffered mode, this interrupts will not be generated, even if it is enabled.
7	Reserved	0	Read returns 0. Writes have no effect.
6	RXOVRNINTENA	0	Overrun interrupt enable. Overrun interrupt will not be generated.
		1	Overrun interrupt will be generated.
5	Reserved	0	Read returns 0. Writes have no effect.



**Table 23-10. SPI Interrupt Register (SPIINT0) Field Descriptions (continued)**

Bit	Field	Value	Description
4	BITERRENA		Enables interrupt on bit error.
		0	No interrupt asserted upon bit error.
		1	Enables interrupt on bit error.
3	DESYNCEN A		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	PARERREN A		Enables interrupt-on-parity-error.
		0	No interrupt asserted on parity error.
		1	An interrupt is asserted on a parity error.
1	TIMEOUTEN A		Enables interrupt on ENA signal time-out.
		0	No interrupt asserted upon ENA signal time-out.
		1	An interrupt is asserted on a time-out of the ENA signal.
0	DLENERREN A		Data length error interrupt enable. A data length error occurs under the following conditions.
			<b>Master:</b> When $\overline{\text{SPIENA}}$ is used, if the $\overline{\text{SPIENA}}$ pin from the slave is deasserted before the master has completed its transfer, the data length error is set. That is, if the character length counter has not overflowed while $\overline{\text{SPIENA}}$ deassertion is detected, then it means that the slave has neither received full data from the master nor has it transmitted complete data.
			<b>Slave:</b> When $\overline{\text{SPISCS}}$ pins are used, if the incoming valid $\overline{\text{SPISCS}}$ 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.

### 23.9.4 SPI Interrupt Level Register (SPILVL)

**Figure 23-29. SPI Interrupt Level Register (SPILVL) [offset = 0Ch]**

31	Reserved												16	
R-0														
15	Reserved			10	9	8	7	6	5	4	3	2	1	0
	Reserved			TXINT LVL	RXINT LVL	Reserv ed	RXOV R NINTL	Reserv ed	BITER RLVL	DESY NCLV L	PARE RRLV L	TIMEO UTLVL	DLEN ERRL VL	
	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/W-0

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

**Table 23-11. SPI Interrupt Level Register (SPILVL) Field Descriptions**

Bit	Field	Value	Description
31-10	Reserved	0	Read returns 0. Writes have no effect.
9	TXINTLVL	0	Transmit interrupt level. Transmit interrupt is mapped to interrupt line INT0.
		1	Transmit interrupt is mapped to interrupt line INT1.
8	RXINTLVL	0	Receive interrupt level. Receive interrupt is mapped to interrupt line INT0.
		1	Receive interrupt is mapped to interrupt line INT1.
7	Reserved	0	Read returns 0. Writes have no effect.
6	RXOVRNINTLVL	0	Receive overrun interrupt level. Receive overrun interrupt is mapped to interrupt line INT0.
		1	Receive overrun interrupt is mapped to interrupt line INT1.
5	Reserved	0	Read returns 0. Writes have no effect.
4	BITERRLVL	0	Bit error interrupt level. Bit error interrupt is mapped to interrupt line INT0.
		1	Bit error interrupt is mapped to interrupt line INT1.
3	DESYNCLVL	0	Desynchronized slave interrupt level. (master mode only). An interrupt caused by desynchronization of the slave is mapped to interrupt line INT0.
		1	An interrupt caused by desynchronization of the slave is mapped to interrupt line INT1.
2	PARERRLVL	0	Parity error interrupt level. A parity error interrupt is mapped to interrupt line INT0.
		1	A parity error interrupt is mapped to interrupt line INT1.
1	TIMEOUTLVL	0	SPIEN $\bar{A}$ pin time-out interrupt level. 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	DLENERLVL	0	Data length error interrupt level (line) select. 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.

### 23.9.5 SPI Flag Register (SPIFLG)

SPIFLAG bits are cleared on read. Software must check all flag bits when reading this register.

**Figure 23-30. SPI Flag Register (SPIFLG) [offset = 10h]**

31						25	24	23						16											
Reserved					R-0	BUF INIT ACTIV E		R-0	Reserved					R-0											
					15			10	9	8	7	6	5	4	3	2	1	0							
Reserved					R-0	TXINT FLG	R-0	RXINT FLG	R/WC- 0	Reserv ed	R-0	RXOV RNINT FLG	R/WC- 0	Reserv ed	R-0	BITER RFLG	R/WC- 0	DESY NCFL G	R/WC- 0	PARE RRFL G	R/WC- 0	TIMEO UT FLG	R/WC- 0	DLEN ERR FLG	R/WC- 0

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

**Table 23-12. SPI Flag Register (SPIFLG) Field Descriptions**

Bit	Field	Value	Description
31-25	Reserved	0	Read returns 0. Writes have no effect.
24	BUFINITACTI VE	0 1	<p>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.</p> <p><b>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.</b></p> <p>0 Multi-buffer RAM initialization is complete.</p> <p>1 Multi-buffer RAM is still being initialized. Do not attempt to write to either multi-buffer RAM or any multi-buffer mode registers.</p>
23-10	Reserved	0	Read returns 0. Writes have no effect.
9	TXINTFLG	0 1	<p>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:</p> <ul style="list-style-type: none"> <li>Writing a new data to either SPIDAT0 or SPIDAT1</li> <li>Writing a 0 to SPIEN (SPIGCR1[24])</li> </ul> <p>0 Transmit buffer is now full. No interrupt pending for transmitter empty.</p> <p>1 Transmit buffer is empty. An interrupt is pending to fill the transmitter.</p>
8	RXINTFLG		<p>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:</p> <ul style="list-style-type: none"> <li>Reading the SPIBUF register</li> <li>Reading TGINTVECT0 or TGINTVECT1 register when there is a receive buffer full interrupt</li> <li>Writing a 1 to this bit</li> <li>Writing a 0 to SPIEN (SPIGCR1[24])</li> <li>System reset</li> </ul> <p>During emulation mode, however, a read to the emulation register (SPIEMU) does not clear this flag bit.</p> <p>No new received data pending. Receive buffer is empty.</p> <p>A newly received data is ready to be read. Receive buffer is full.</p> <p><b>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.</b></p>

**Table 23-12. SPI Flag Register (SPIFLG) Field Descriptions (continued)**

Bit	Field	Value	Description
7	Reserved	0	Read returns 0. Writes have no effect.
6	RXOVRNINT FLG	0	<p>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:</p> <ul style="list-style-type: none"> <li>• Reading TGINTVECT0 or TGINTVECT1 register when there is a receive-buffer-overrun interrupt</li> <li>• Writing a 1 to RXOVRNINTFLG in the SPI Flag Register (SPIFLG) register itself</li> <li>• Writing a 0 to SPIEN</li> <li>• Reading the data field of the SPIBUF register</li> </ul> <p><b>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.</b></p> <p><b>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 RXOVRN 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.</b></p> <p>In multi-buffer mode of MibSPI, this bit is cleared under the following conditions.</p> <ul style="list-style-type: none"> <li>• Reading the RXOVRN_BUF_ADDR register</li> <li>• Writing a 1 to RXOVRNINTFLG in the SPI Flag Register (SPIFLG) register itself</li> </ul> <p>In multi-buffer mode, if RXOVRNINTFLG is set, then the address of the buffer which experienced the overrun is available in RXOVRN_BUF_ADDR.</p> <p>Overrun condition did not occur.</p> <p>Overrun condition has occurred.</p>
5	Reserved	0	Read returns 0. Writes have no effect.
4	BITERRFLG	0 1	<p>Mismatch of internal transmit data and transmitted data. This flag can be cleared by one of the following methods.</p> <ul style="list-style-type: none"> <li>• Write a 1 to this bit.</li> <li>• Set the SPIENA bit to 0.</li> </ul> <p>0 No bit error occurred.</p> <p>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.</p>
3	DESYNCFLG	0 1	<p>Desynchronization of slave device. Desynchronization monitor is active in master mode only.</p> <p>0 No slave desynchronization detected.</p> <p>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 <math>t_{ZDEDELAY}</math>. If DESYNCENA is set an interrupt is asserted. Desynchronization can occur if a slave device misses a clock edge coming from the master.</p> <p>This flag can be cleared by one of the following methods.</p> <ul style="list-style-type: none"> <li>• Write a 1 to this bit.</li> <li>• Set SPIENA bit to 0.</li> </ul>
2	PARITYERR FLG	0 1	<p>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.</p> <p>0 No parity error detected.</p> <p>1 A parity error occurred.</p> <p>This flag can be cleared by one of the following methods.</p> <ul style="list-style-type: none"> <li>• Write a 1 to this bit.</li> <li>• Set SPIENA bit to 0.</li> </ul>

**Table 23-12. SPI Flag Register (SPIFLG) Field Descriptions (continued)**

Bit	Field	Value	Description
1	TIMEOUTFLAG	0 1	<p>Time-out caused by nonactivation of ENA signal.</p> <p>No ENA-signal time-out occurred.</p> <p>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.</p> <p>This flag can be cleared by one of the following methods.</p> <ul style="list-style-type: none"> <li>• Write a 1 to this bit.</li> <li>• Set SPIENA bit to 0.</li> </ul>
0	DLENERRFLAG	0 1	<p>Data-length error flag.</p> <p>This flag can be cleared by one of the following methods.</p> <ul style="list-style-type: none"> <li>• Write a 1 to this bit.</li> <li>• Set SPIENA bit to 0.</li> </ul> <p><b>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.</b></p> <p>No data length error has occurred.</p> <p>A data length error has occurred.</p>

### 23.9.6 SPI Pin Control Register 0 (SPIPC0)

**NOTE: Register bits vary by device**

Register bits 31:24 and 23:16 of SPIPC0 to SPIPC9 reflect the number of SIMO/SOMI data lines per device. On devices with 8 data-line support, all of bits 31 to 16 are implemented. On devices with less than 8 data lines, only a subset of these bits are available. Unimplemented bits return 0 upon read and are not writable.

**Figure 23-31. SPI Pin Control Register 0 (SPIPC0) [offset = 14h]**

31									24	23	16				
SOMIFUN												SIMOFUN			
R/W-0												R/W-0			
15	12		11	10	9	8	7					0			
Reserved			SOMIFUN0	SIMOFUN0	CLKFUN	ENAFUN	SCSFUN								
R-0			R/W-0	R/W-0	R/W-0	R/W-0U	R/W-0U								

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

**Table 23-13. SPI Pin Control (SPIPC0) Field Descriptions**

Bit	Field	Value	Description
31-24	SOMIFUN	0 1	<p>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.</p> <p><b>Note: Duplicate Control Bits for SPISOMI[x]. 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[x] pin. The read value of Bit 24 always reflects the value of bit 11.</b></p> <p>0 SPISOMI[x] pin is a GIO pin 1 SPISOMI[x] pin is a SPI functional pin</p>
23-16	SIMOFUN	0 1	<p>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.</p> <p><b>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 SPISIMO[x] pin. The read value of Bit 16 always reflects the value of bit 10.</b></p> <p>0 The SPISIMOx pin is a GIO pin. 1 The SPISIMOx pin is a SPI functional pin</p>
15-12	Reserved	0	Read returns 0. Writes have no effect.
11	SOMIFUN0	0 1	<p>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.</p> <p>0 The SPISOMI0 pin is a GIO pin. 1 The SPISOMI0 pin is a SPI functional pin.</p> <p><b>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.</b></p>
10	SIMOFUN0	0 1	<p>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.</p> <p>0 The SPISIMO0 pin is a GIO pin. 1 The SPISIMO0 pin is a SPI functional pin.</p> <p><b>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.</b></p>
9	CLKFUN	0 1	<p>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.</p> <p>0 The SPICLK pin is a GIO pin. 1 The SPICLK pin is a SPI functional pin.</p>

**Table 23-13. SPI Pin Control (SPIPC0) Field Descriptions (continued)**

Bit	Field	Value	Description
8	ENAFUN	0 1	<p><math>\overline{\text{SPIENA}}</math> function. This bit determines whether the <math>\overline{\text{SPIENA}}</math> pin is to be used as a general-purpose I/O pin or as a SPI functional pin.</p> <p>0 The <math>\overline{\text{SPIENA}}</math> pin is a GIO pin.</p> <p>1 The <math>\overline{\text{SPIENA}}</math> pin is a SPI functional pin.</p>
7-0	SCSFUN	0 1	<p><math>\overline{\text{SPISCSx}}</math> function. Determines whether each <math>\overline{\text{SPISCSx}}</math> pin is to be used as a general-purpose I/O pin or as a SPI functional pin. If the slave <math>\overline{\text{SPISCSx}}</math> pins are in functional mode and receive an inactive high signal, the slave SPI will place its output in high-z and disable shifting.</p> <p>0 The <math>\overline{\text{SPISCSx}}</math> pin is a GIO pin.</p> <p>1 The <math>\overline{\text{SPISCSx}}</math> pin is a SPI functional pin.</p>

### 23.9.7 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 23-32. SPI Pin Control Register 1 (SPIPC1) [offset = 18h]**

31							24	23	16
SOMIDIR				SIMODIR					
R/W-0				R/W-0					
15	12	11	10	9	8	7	0		
Reserved		SOMI DIR0	SIMO DIR0	CLKDI R	ENADI R	SCSDIR			
R-0		R/W-0	R/W-0	R/W-0	R/W- 0U	R/W-0U			

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

**Table 23-14. SPI Pin Control Register (SPIPC1) Field Descriptions**

Bit	Field	Value	Description
31-24	SOMIDIR		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.
		0	SPISOMIx pin is an input
		1	SPISOMIx pin is an output
23-16	SIMODIR		SPISIMIOx direction. Controls the direction of SPISIMIOx when used for general-purpose I/O. If SPISIMIOx pin is used as a SPI functional pin, the I/O direction is determined by the MASTER bit in the SPIGCR1 register.
		0	SPISIMIOx pin is an input
		1	SPISIMIOx pin is an output
15-12	Reserved	0	Read returns 0. Writes have no effect.
11	SOMIDIR 0		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	The SPISOMI0 pin is an input.
		1	The SPISOMI0 pin is an output.
10	SIMODIR 0		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	The SPISIMO0 pin is a GIO pin.
		1	The SPISIMO0 pin is a SPI functional pin.
9	CLKDIR		SPICLK direction. This bit controls the direction of the SPICLK pin when it is used as a general-purpose I/O pin. In functional mode, the I/O direction is determined by the CLKMOD bit.)
		0	The SPICLK pin is an input.
		1	The SPICLK pin is an output.



**Table 23-14. SPI Pin Control Register (SPIPC1) Field Descriptions (continued)**

Bit	Field	Value	Description
8	ENADIR	0 1	<p><math>\overline{\text{SPIENA}}</math> direction. This bit controls the direction of the <math>\overline{\text{SPIENA}}</math> pin when it is used as a general-purpose I/O. If the <math>\overline{\text{SPIENA}}</math> pin is used as a functional pin, then the I/O direction is determined by the CLKMOD bit (SPIGCR1[1]).</p> <p>0 The <math>\overline{\text{SPIENA}}</math> pin is an input.</p> <p>1 The <math>\overline{\text{SPIENA}}</math> pin is an output.</p>
7-0	SCSDIR	0 1	<p><math>\overline{\text{SPISCSx}}</math> direction. These bits control the direction of each <math>\overline{\text{SPISCSx}}</math> pin when it is used as a general-purpose I/O pin. Each pin could be configured independently from the others. If the <math>\overline{\text{SPISCSx}}</math> is used as a SPI functional pin; the I/O direction is determined by the CLKMOD bit (SPIGCR1[1]).</p> <p>0 The <math>\overline{\text{SPISCSx}}</math> pin is an input.</p> <p>1 The <math>\overline{\text{SPISCSx}}</math> pin is an output.</p>

### 23.9.8 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 23-33. SPI Pin Control Register 2 (SPIPC2) [offset = 1Ch]**

31	SOMIDIN								24	23	SIMODIN								16
	R/W-U										R/W-U								
15	Reserved				12	11	10	9	8	7	SCSDIN				0				
	R-0					R-U	R-U	R-U	R-U		R/W-U								

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

**Table 23-15. SPI Pin Control Register 2 (SPIPC2) Field Descriptions**

Bit	Field	Value	Description
31-24	SOMIDIN	0 1	SPISOMIx data in. The value of the SPISOMIx pins. The SPISOMIx pin is logic 0. The SPISOMIx pin is logic 1.
23-16	SIMODIN	0 1	SPISIMOX data in. The value of the SPISIMOX pins. The SPISIMOX pin is logic 0. The SPISIMOX pin is logic 1.
15-12	Reserved	0	Read returns 0. Writes have no effect.
11	SOMIDIN <sub>0</sub>	0 1	SPISOMI0 data in. The value of the SPISOMI0 pin. The SPISOMI0 pin is logic 0. The SPISOMI0 pin is logic 1.
10	SIMODIN <sub>0</sub>	0 1	SPISIMO0 data in. The value of the SPISIMO0 pin. The SPISIMO0 pin is logic 0. The SPISIMO0 pin is logic 1.
9	CLKDIN	0 1	Clock data in. The value of the SPICLK pin. pin. The SPICLK pin is logic 0. The SPICLK pin is logic 1.
8	ENADIN	0 1	$\overline{\text{SPIEN}}_A$ data in. The the value of the $\overline{\text{SPIEN}}_A$ pin. The $\overline{\text{SPIEN}}_A$ pin is logic 0. The $\overline{\text{SPIEN}}_A$ pin is logic 1.
7-0	SCSDIN	0 1	$\overline{\text{SPISCS}}_x$ data in. The value of the $\overline{\text{SPISCS}}_x$ pin. The $\overline{\text{SPISCS}}_x$ pin is logic 0. The $\overline{\text{SPISCS}}_x$ pin is logic 1.

### 23.9.9 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 23-34. SPI Pin Control Register 3 (SPIPC3) [offset = 20h]**

31											24	23											16								
SOMIDOUT															SIMODOUT																
R/W-U															R/W-U																
15															12	11	10	9	8	7											0
Reserved												SOMI DOUT	SIMO DOUT	CLKD OUT	ENAD OUT	SCSDOUT															
R-0												0	0	R/W-U	R/W-U	R/W-U	R/W-U														

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

**Table 23-16. SPI Pin Control Register 3 (SPIPC3) Field Descriptions**

Bit	Field	Value	Description
31-24	SOMIDOUT	0 1	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.  <b>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.</b> 0 Current value on SPISOMIx pin is logic 0. 1 Current value on SPISOMIx pin is logic 1
23-16	SIMODOUT	0 1	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.  <b>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.</b> 0 Current value on SPISIMOX pin is logic 0. 1 Current value on SPISIMOX pin is logic 1.
15-12	Reserved	0	Read returns 0. Writes have no effect.
11	SOMIDOUT0	0 1	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	0 1	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	0 1	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	0 1	SPIEN $\bar{A}$ data out write. Only active when the SPIEN $\bar{A}$ 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 SPIEN $\bar{A}$ pin is logic 0. 1 The SPIEN $\bar{A}$ pin is logic 1.

**Table 23-16. SPI Pin Control Register 3 (SPIPC3) Field Descriptions (continued)**

Bit	Field	Value	Description
7-0	SCSDOUT		<p><math>\overline{\text{SPISCSx}}</math> data out write. Only active when the <math>\overline{\text{SPISCSx}}</math> 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.</p>
		0	The $\overline{\text{SPISCSx}}$ pin is logic 0.
		1	The $\overline{\text{SPISCSx}}$ pin is logic 1.

### 23.9.10 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 23-35. SPI Pin Control Register 4 (SPIPC4) [offset = 24h]**

31							24	23							16
SOMISET						SIMOSET									
R/W-U						R/W-U									
15			12	11	10	9	8	7					0		
Reserved			SOMI SET0	SIMO SET0	CLKS ET	ENAS ET	SCSSET								
R-0			R/W-U	R/W-U	R/W-U	R/W-U	R/W-U								

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

**Table 23-17. SPI Pin Control Register 4 (SPIPC4) Field Descriptions**

Bit	Field	Value	Description
31-24	SOMISET	0 1	SPISOMIx data out set. This pin is only active when the SPISOMIx pin is configured as a general-purpose output pin. <b>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.</b> Read: SPISIMOX is logic 0. Write: Writing a 0 to this bit has no effect. Read: SPISOMIx is logic 1. Write: Logic 1 is placed on SPISOMIx pin if it is in general-purpose output mode.
23-16	SIMOSET	0 1	SPISIMOX data out set. This bit is only active when the SPISIMOX pin is configured as a general-purpose output pin. <b>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.</b> Read: SPISIMIx is logic 0. Write: Writing a 0 to this bit has no effect. Read: SPISIMIx is logic 1. Write: Logic 1 is placed on SPISIMIx pin if it is in general-purpose output mode.
15-12	Reserved	0	Read returns 0. Writes have no effect.
11	SOMISET0	0 1	SPISOMI0 data out set. This pin is only active when the SPISOMI0 pin is configured as a general-purpose output pin. Read: SPISOMI0 is logic 0. Write: Writing a 0 to this bit has no effect. Read: SPISOMI0 is logic 1. Write: Logic 1 is placed on SPISOMI0 pin if it is in general-purpose output mode.
10	SIMOSET0	0 1	SPISIMO0 data out set. This pin is only active when the SPISIMO0 pin is configured as a general-purpose output pin. Read: SPISIMO0 is logic 0. Write: Writing a 0 to this bit has no effect. Read: SPISIMO0 is logic 1. Write: Logic 1 is placed on SPISIMO0 pin if it is in general-purpose output mode.

**Table 23-17. SPI Pin Control Register 4 (SPIPC4) Field Descriptions (continued)**

Bit	Field	Value	Description
9	CLKSET	0 1	<p>SPICLK data out set. This bit is only active when the SPICLK pin is configured as a general-purpose output pin.</p> <p>Read: SPICLK is logic 0. Write: Writing a 0 to this bit has no effect.</p> <p>Read: SPICLK is logic 1. Write: Logic 1 is placed on SPICLK pin if it is in general-purpose output mode.</p>
8	ENASET	0 1	<p><math>\overline{\text{SPIEN}}\overline{\text{A}}</math> data out set. This bit is only active when the <math>\overline{\text{SPIEN}}\overline{\text{A}}</math> pin is configured as a general-purpose output pin.</p> <p>Read: The <math>\overline{\text{SPIEN}}\overline{\text{A}}</math> pin is logic 0. Write: Writing a 0 to this bit has no effect.</p> <p>Read: The <math>\overline{\text{SPIEN}}\overline{\text{A}}</math> pin is logic 1. Write: Logic 1 is placed on <math>\overline{\text{SPIEN}}\overline{\text{A}}</math> pin if it is in general-purpose O/P mode.</p>
7-0	SCSSET	0 1	<p><math>\overline{\text{SPISCS}}\overline{\text{x}}</math> data out set. This bit is only active when the <math>\overline{\text{SPISCS}}\overline{\text{x}}</math> pin is configured as a general-purpose output pin. A value of 1 written to this bit sets the corresponding SCSDOUT bit to 1.</p> <p>Read: The <math>\overline{\text{SPISCS}}\overline{\text{x}}</math> pin is logic 0. Write: Writing a 0 to this bit has no effect.</p> <p>Read: The <math>\overline{\text{SPISCS}}\overline{\text{x}}</math> pin is logic 1. Write: Logic 1 placed on <math>\overline{\text{SPISCS}}\overline{\text{x}}</math> pin if it is in general-purpose output mode.</p>

### 23.9.11 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 23-36. SPI Pin Control Register 5 (SPIPC5) [offset = 28h]**

31							24	23							16
SOMICLR						SIMOCLR									
R/W-U						R/W-U									
15	12		11	10	9	8	7							0	
Reserved			SOMI CLR0	SIMO CLR0	CLKC LR	ENAC LR	SCSCLR								
R-0			R/W-U	R/W-U	R/W-U	R/W- 0U	R/W-0U								

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

**Table 23-18. SPI Pin Control Register 5 (SPIPC5) Field Descriptions**

Bit	Field	Value	Description
31-24	SOMICLR		SPISOMIx data out clear. This pin is only active when the SPISOMIx pin is configured as a general-purpose output pin.  <b>Bit 11 or bit 24 can be used to set the SOMIO pin. If a 32-bit write is performed, bit 11 will have priority over bit 24.</b>  0 Read: The current value on SOMIDOUTx is 0. Write: Writing a 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		SPISIMOX data out clear. This bit is only active when the SPISIMOX pin is configured as a general-purpose output pin.  <b>Bit 10 or bit 16 can be used to set the SOMIO pin. If a 32-bit write is performed, bit 10 will have priority over bit 16.</b>  0 Read: The current value on SOMODOUTx is 0. Write: Writing a 0 to this bit has no effect.  1 Read: The current value on SOMODOUTx is 1. Write: Logic 0 is placed on SPISIMIx pin if it is in general-purpose output mode.
15-12	Reserved	0	Read returns 0. Writes have no effect.
11	SOMICLR0		SPISOMIO data out clear. This pin is only active when the SPISOMIO pin is configured as a general-purpose output pin.  0 Read: The current value on SPISOMIO is 0. Write: Writing a 0 to this bit has no effect.  1 Read: The current value on SPISOMIO is 1. Write: Logic 0 is placed on SPISOMIO pin if it is in general-purpose output mode.
10	SIMOCLR0		SPISIMO0 data out clear. This pin 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: Writing a 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.

**Table 23-18. SPI Pin Control Register 5 (SPIPC5) Field Descriptions (continued)**

Bit	Field	Value	Description
9	CLKCLR	0 1	<p>SPICLK data out clear. This bit is only active when the SPICLK pin is configured as a general-purpose output pin.</p> <p>Read: The current value on SPICLK is 0. Write: Writing a 0 to this bit has no effect.</p> <p>Read: The current value on SPICLK is 1. Write: Logic 0 is placed on SPICLK pin if it is in general-purpose output mode.</p>
8	ENACLK	0 1	<p><math>\overline{\text{SPIEN}}_A</math> data out clear. This bit is only active when the <math>\overline{\text{SPIEN}}_A</math> pin is configured as a general-purpose output pin. A value of 1 written to this bit clears the corresponding ENABLEDOUT bit to 0.</p> <p>Read: The current value on ENA is 0. Write: Writing a 0 to this bit has no effect.</p> <p>Read: The current value on ENA is 1. Write: Logic 0 is placed on <math>\overline{\text{SPIEN}}_A</math> pin if it is in general-purpose O/P mode.</p>
7-0	SCSCLR	0 1	<p><math>\overline{\text{SPISCS}}_x</math> data out clear. This bit is only active when the <math>\overline{\text{SPISCS}}_x</math> pin is configured as a general-purpose output pin.</p> <p>Read: The current value on SCSDOUT<sub>x</sub> is 0. Write: Writing a 0 to this bit has no effect.</p> <p>Read: The current value on SCSDOUT<sub>x</sub> is 1. Write: Logic 0 placed on <math>\overline{\text{SPISCS}}_x</math> pin if it is in general-purpose output mode.</p>



### 23.9.12 SPI Pin Control Register 6 (SPIPC6)

**NOTE: Register bits vary by device**

Register bits 31:24 and 23:16 of SPIPC0 to SPIPC9 reflect the number of SIMO/SOMI data lines per device. On devices with 8 data-line support, all of bits 31 to 16 are implemented. On devices with less than 8 data lines, only a subset of these bits are available. Unimplemented bits return 0 upon read and are not writable.

**Figure 23-37. SPI Pin Control Register 6 (SPIPC6) [offset = 2Ch]**

31							24	23							16
SOMIPDR						SIMOPDR									
R/W-0						R/W-0									
15			12	11	10	9	8	7					0		
Reserved			SOMI PDR0	SIMO PDR0	CLKP DR	ENAP DR	SCSPDR								
R-0			R/W-0	R/W-0	R/W-0	R/W-0	R/W-0								

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

**Table 23-19. SPI Pin Control Register 6 (SPIPC6) Field Descriptions**

Bit	Field	Value	Description
31-24	SOMIPDR	0 1	<p>SPISOMIx open drain enable. This bit enables open drain capability for the SPISOMIx pin if the following conditions are met:</p> <ul style="list-style-type: none"> <li>SOMIDIRx = 1 (SPISOMIx pin configured in GIO mode as an output)</li> <li>SOMIDOUTx = 1</li> </ul> <p><b>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.</b></p> <p>0 The output value on the SPISOMIx pin is logic 1. 1 Output pin SPISOMIx is in a high-impedance state.</p>
23-16	SIMOPDR	0 1	<p>SPISIMOX open drain enable. This bit enables open drain capability for the SPISIMOX pin if the following conditions are met:</p> <ul style="list-style-type: none"> <li>SIMODIRx = 1 (SPISIMOX pin configured in GIO mode as an output)</li> <li>SIMODOUTx = 1</li> </ul> <p><b>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.</b></p> <p>0 The output value on SPISIMOX pin is logic 1. 1 Output pin SPISIMOX is in a high-impedance state.</p>
15-12	Reserved	0	Read returns 0. Writes have no effect.
11	SOMIPDR0	0 1	<p>SOMI0 open-drain enable. This bit enables open-drain capability for SOMI0 if the following conditions are met:</p> <ul style="list-style-type: none"> <li>SOMI0 pin configured in GIO mode as output pin</li> <li>Output value on SPISOMI0 pin is logic 1.</li> </ul> <p>0 Output value 1 of SPISOMI0 pin is logic 1. 1 Output value 1 of SPISOMI0 is high-impedance.</p>
10	SIMOPDR0	0 1	<p>SPISIMO0 open-drain enable. This bit enables open -drain capability for the SPISIMO0 pin if the following conditions are met:</p> <ul style="list-style-type: none"> <li>SIMO0 pin configured in GIO mode as output pin</li> <li>Output value on SPISIMO0 pin is logic 1.</li> </ul> <p>0 Output value 1 of SPISIMO0 pin is logic 1. 1 Output value 1 of SPISIMO0 is high-impedance.</p>

**Table 23-19. SPI Pin Control Register 6 (SPIPC6) Field Descriptions (continued)**

Bit	Field	Value	Description
9	CLKPDR	0 1	<p>CLK open drain enable. This bit enables open drain capability for the pin CLK if the following conditions are met:</p> <ul style="list-style-type: none"> <li>• SPICLK pin configured in GIO mode as an output pin</li> <li>• SPICLKDOUT = 1</li> </ul> <p>Output value on CLK pin is logic 1.</p> <p>Output pin CLK is in a high-impedance state.</p>
8	ENAPDR	0 1	<p>SPIENA pin open drain enable. This bit enables open drain capability for SPIENA if the following conditions are met:</p> <ul style="list-style-type: none"> <li>• SPIENA pin configured in GIO mode as an output pin</li> <li>• SPIENADOUT = 1</li> </ul> <p>Output value SPIENA on pin is logic 1.</p> <p>Output pin SPIENA is in a high-impedance state.</p>
7-0	SCSPDR	0 1	<p>SPISCSx open drain enable. This bit enables open drain capability for the SPISCSx pin if the following conditions are met:</p> <ul style="list-style-type: none"> <li>• SPISCS pin configured in GIO mode as an output pin</li> <li>• SPISCSDOUTx = 1</li> </ul> <p>Output value on SCSx pin is logic 1.</p> <p>Output pin SCSx is in a high-impedance state.</p>

### 23.9.13 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 23-38. SPI Pin Control Register 7 (SPIPC7) [offset = 30h]**

31					24	23					16
SOMIDIS						SIMODIS					
R/W-n						R/W-n					
15			12	11	10	9	8	7			0
Reserved			SOMI PDIS0	SIMO PDIS0	CLKP DIS	ENAP DIS	SCSPDIS				
R-0			R/W-n	R/W-n	R/W-n	R/W- nU	R/W-nU				

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

**Table 23-20. SPI Pin Control Register 7 (SPIPC7) Field Descriptions**

Bit	Field	Value	Description
31-24	SOMIDIS	0 1	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. <b>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.</b> Pull control on the SPISOMIx pin is enabled. Pull control on the SPISOMIx pin is disabled.
23-16	SIMODIS	0 1	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. <b>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.</b> Pull control on SPISIMOX pin is enabled. Pull control on SPISIMOX pin is disabled.
15-12	Reserved	0	Read returns 0. Writes have no effect.
11	SOMIPDIS0	0 1	SPISOMIO pull control enable/disable. This bit enables pull control capability for the pin SPISOMIO pin if it is in input mode regardless of whether it is in functional or GIO mode. Pull control on the SPISOMIO pin is enabled. Pull control on the SPISOMIO pin is disabled.
10	SIMOPDIS0	0 1	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. Pull control on the SPISIMO0 pin is enabled. Pull control on the SPISIMO0 pin is disabled.
9	CLKPDIS	0 1	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. Pull control on the CLK pin is enabled. Pull control on the CLK pin is disabled.

**Table 23-20. SPI Pin Control Register 7 (SPIPC7) Field Descriptions (continued)**

Bit	Field	Value	Description
8	ENAPDIS		ENABLE pull control enable/disable. This bit enables pull control capability for the pin $\overline{\text{SPIENA}}$ 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		SCSx pull control enable/disable. This bit enables pull control capability for the pin $\overline{\text{SPISCSx}}$ 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.

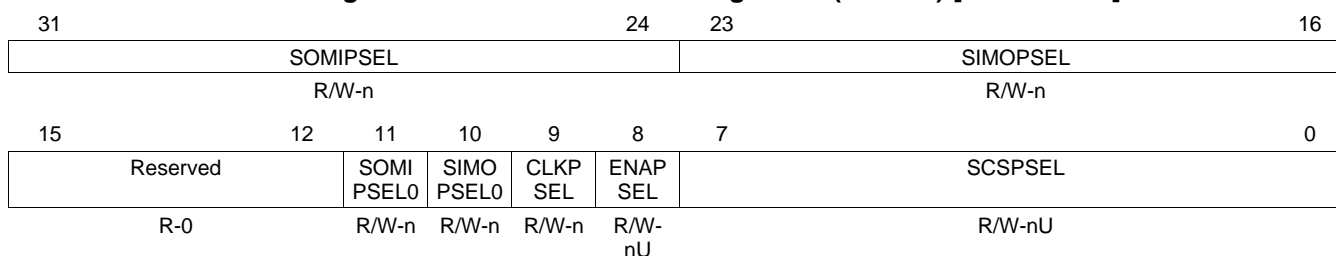
### 23.9.14 SPI Pin Control Register 8 (SPIPC8)

**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 23-39. SPI Pin Control Register 8 (SPIPC8) [offset = 34h]**


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

**Table 23-21. SPI Pin Control Register 8 (SPIPC8) Field Descriptions**

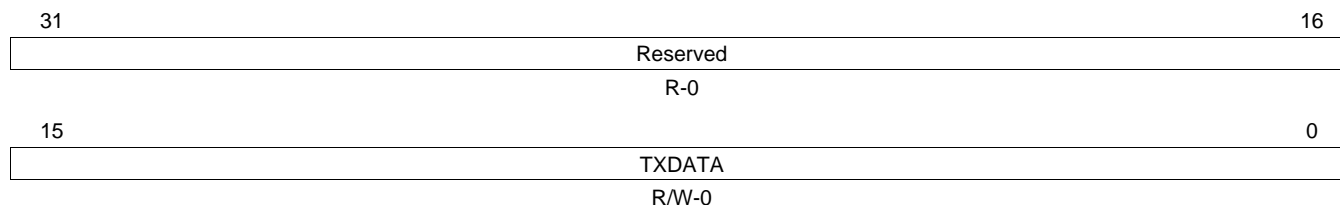
Bit	Field	Value	Description
31-24	SOMIPSEL	0 1	SPISOMIx pull select. This bit selects the type of pull logic at the SOMIx pin. <b>Note: Bit 11 or bit 24 can be used to set pull-select for SPISOMI0. If a 32-bit write is performed, bit 11 will have priority over bit 24.</b> 0 Pull down on SOMIx pin 1 Pull up on SOMIx pin
23-16	SIMOPSEL	0 1	SPISIMOX pull select. This bit selects the type of pull logic at the SPISIMOX pin. <b>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.</b> 0 Pull down on SPISIMOX pin 1 Pull up on SPISIMOX pin
15-12	Reserved	0	Read returns 0. Writes have no effect.
11	SOMIPSEL0	0 1	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 SPISOMI pin
10	SIMOPSEL0	0 1	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	0 1 0 1	CLK pull select. This bit selects the type of pull logic at the CLK pin. 0 Pull down on CLK pin 1 Pull down on CLK pin 0 Input buffer for CLK is disabled if PULLDIS = 1 1 Input buffer for CLK is enabled if PULLDIS = 1

**Table 23-21. SPI Pin Control Register 8 (SPIPC8) Field Descriptions (continued)**

Bit	Field	Value	Description
8	ENAPSE L	0	ENABLE pull select. This bit selects the type of pull logic at the ENABLE pin. Pull down on ENABLE pin
		1	Pull down on ENABLE pin
7-0	SCSPSE L	0	SCSx pull select. This bit selects the type of pull logic at the SCSx pin. Pull down on SCSx pin
		1	Pull down on SCSx pin

### 23.9.15 SPI Transmit Data Register 0 (SPIDAT0)

Figure 23-40. SPI Transmit Data Register 0 (SPIDAT0) [offset = 38h]



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

Table 23-22. SPI Transmit Data Register 0 (SPIDAT0) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	TXDATA	0-FFFFh	<p><b>SPI transmit data.</b> When written, these bits will be copied to the shift register if it is empty. If the shift register is not empty, TXBUF holds the written data. SPIEN (SPICGR1[24]) must be set to 1 before this register can be written to. Writing a 0 to the SPIEN register forces the lower 16 bits of the SPIDAT0 to 0x00.</p> <p><b>Note: When this register is read, the contents TXBUF, which holds the latest written data, will be returned.</b></p> <p><b>Note: Regardless of character length, the transmit word should be right-justified before writing to the SPIDAT1 register.</b></p> <p><b>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.</b></p>

**23.9.16 SPI Transmit Data Register 1 (SPIDAT1)**
**Figure 23-41. SPI Transmit Data Register 1 (SPIDAT1) [offset = 3Ch]**

31	29	28	27	26	25	24	23	16
Reserved	CSHOLD	Rsvd.	WDEL	DFSEL				CSNR
R-0	R/W-0	R-0	R/W-0	R/W-0				R/W-0
15	TXDATA							0
								R/W-0

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

**Table 23-23. SPI Transmit Data Register 1 (SPIDAT1) Field Descriptions**

Bit	Field	Value	Description
31-29	Reserved	0	Read returns 0. Writes have no effect.
28	CSHOLD	0 1	<p>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.</p> <p>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.</p> <p>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.</p>
27	Reserved	0	Reads returns 0. Writes have no effect.
26	WDEL	0 1	<p>Enable the delay counter at the end of the current transaction.</p> <p><b>Note: The WDEL bit is supported in master mode only. In slave mode, this bit will be ignored.</b></p> <p>0 No delay will be inserted. However, SPISCS pins will still be de-activated for at least for 2VCLK cycles if CSHOLD = 0.</p> <p><b>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).</b></p> <p>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 SPISCS pins will be de-activated for at least (WDELAY + 2) * VCLK_Period duration.</p>
25-24	DFSEL	0 1h 2h 3h	<p>Data word format select</p> <p>0 Data word format 0 is selected</p> <p>1h Data word format 1 is selected</p> <p>2h Data word format 2 is selected</p> <p>3h Data word format 3 is selected</p>
23-16	CSNR	0-FFh	<p>Chip select number. CSNR defines the chip-select that will be activated during the data transfer.</p> <p><b>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.</b></p>
15-0	TXDATA	0-FFFFh	<p><b>Transfer data.</b> 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.</p> <p>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.</p> <p>Write to this register ONLY when using the automatic slave chip-select feature (see <a href="#">Section 23.2</a>). 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.</p> <p>When this register is read, the contents of TXBUF, which holds the latest data written, will be returned.</p> <p><b>Note: Regardless of the character length, the transmit data should be right-justified before writing to the SPIDAT1 register.</b></p>



### 23.9.17 SPI Receive Buffer Register (SPIBUF)

**Figure 23-42. SPI Receive Buffer Register (SPIBUF) [offset = 40h]**

31	30	29	28	27	26	25	24	23			16
RXEM PTY	RXOV R	TXFUL L	BITER R	DESY NC	PARIT YERR	TIMEO UT	DLEN ERR	LCSNR			
R-1	R-0	R-0	R-0	R-0	R-0	R-0	R-0			R-0	
											0
15	RXDATA										0
											R-0

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

**Table 23-24. SPI Receive Buffer Register (SPIBUF) Field Descriptions**

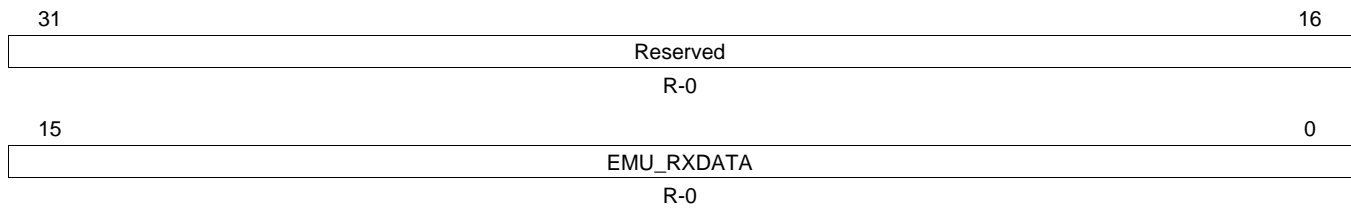
Bit	Field	Value	Description
31	RXEMPTY	0 1	<p><b>Receive data buffer empty.</b>When the host reads the SPIBUF field or the entire SPIBUF register, it automatically sets the RXEMPTY flag. When a data transfer is completed, the received data is copied into SPIBUF, and the RXEMPTY flag is cleared.</p> <p>New data has been received and copied into the SPIBUF field.</p> <p>No data has been received since the last read of SPIBUF.</p> <p>This flag gets set to 1 under the following conditions:</p> <ul style="list-style-type: none"> <li>Reading the RXDATA portion of the SPIBUF register.</li> <li>Writing a 1 to clear the RXINTFLG bit in the SPI Flag Register (SPIFLG) register.</li> </ul> <p>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).</p>
30	RXOVR	0 1	<p>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).</p> <p>If enabled, the RXOVRN interrupt is generated when RXBUF is overwritten, and reading either SPI Flag Register (SPIFLG) or SPIVEXTx 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).</p> <p>This flag is cleared to 0 when the RXDATA is read.</p> <p><b>Note: A special condition under which RXOVR flag gets set. If both SPIBUF and RXBUF are already full and while another buffer receive is underway, if any errors such as TIMEOUT, BITERR and DLEN_ERR occur, then RXOVR in RXBUF and SPI Flag Register (SPIFLG) registers will be set to indicate that the status flags are getting overwritten by the new transfer. This overrun should be treated like a normal receive overrun.</b></p> <p>No receive data overrun condition occurred since last read of the data field.</p> <p>A receive data overrun condition occurred since last read of the data field.</p>
29	TXFULL	0 1	<p><b>Transmit data buffer full.</b>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.</p> <p>The transmit buffer is empty; SPIDAT0/SPIDAT1 is ready to accept a new data.</p> <p>The transmit buffer is full; SPIDAT0/SPIDAT1 is not ready to accept new data.</p>
28	BITERR	0 1	<p><b>Bit error.</b>There was a mismatch of internal transmit data and transmitted data.</p> <p>No bit error occurred.</p> <p><b>Note: This flag is cleared to 0 when the RXDATA portion of the SPIBUF register is read.</b></p> <p>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.</p>

**Table 23-24. SPI Receive Buffer Register (SPIBUF) Field Descriptions (continued)**

Bit	Field	Value	Description
27	DESYNC	0 1	<p><b>Desynchronization of slave device.</b>This bit is valid in master mode only.</p> <p>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 <math>t_{2EDELAY}</math>. If DESYNCENA is set, an interrupt is asserted. Desynchronization can occur if a slave device misses a clock edge coming from the master.</p> <p><b>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.</b></p> <p>0 No slave desynchronization detected.</p> <p><b>Note: This flag is cleared to 0 when the RXDATA portion of the SPIBUF register is read.</b></p> <p>1 A slave device is desynchronized.</p>
26	PARITYERR	0 1	<p><b>Parity error.</b>The calculated parity differs from the received parity bit.</p> <p>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.</p> <p><b>Note: This flag is cleared to 0 when the RXDATA portion of the SPIBUF register is read.</b></p> <p>0 No parity error detected.</p> <p>1 A parity error occurred.</p>
25	TIMEOUT	0 1	<p>Time-out because of non-activation of ENA pin.</p> <p>The SPI generates a time-out when the slave does not respond in time by activating the ENA signal after the chip select signal has been activated. If a time-out condition is detected, the corresponding chip select is deactivated immediately and the TIMEOUT flag is set. In addition, the TIMEOUT flag in the status field of the corresponding buffer and in the SPI Flag Register (SPIFLG) register is set.</p> <p><b>This bit is valid only in master mode.</b></p> <p>This flag is cleared to 0 when RXDATA portion of the SPIBUF register is read.</p> <p>0 No ENA-pin time-out occurred.</p> <p>1 An ENA signal time-out occurred.</p>
24	DLENERR	0 1	<p><b>Data length error flag.</b></p> <p><b>Note: This flag is cleared to 0 when the RXDATA portion of the SPIBUF register is read.</b></p> <p>0 No data-length error has occurred.</p> <p>1 A data length error has occurred.</p>
23-16	LCSNR	0-FFh	<p><b>Last chip select number.</b> 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.</p>
15-0	RXDATA	0-FFFFh	<p><b>SPI receive data.</b> 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.</p>

### 23.9.18 SPI Emulation Register (SPIEMU)

Figure 23-43. SPI Emulation Register (SPIEMU) [offset = 44h]



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

Table 23-25. SPI Emulation Register (SPIEMU) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	EMU_RXD ATA	0-FFFFh	<b>SPI receive data.</b> 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.

**23.9.19 SPI Delay Register (SPIDELAY)**
**Figure 23-44. SPI Delay Register (SPIDELAY) [offset = 48h]**

31	24	23	16
C2TDELAY		T2CDELAY	
R/W-0		R/W-0	
15	8	7	0
T2EDELAY		C2EDELAY	
R/W-0		R/W-0	

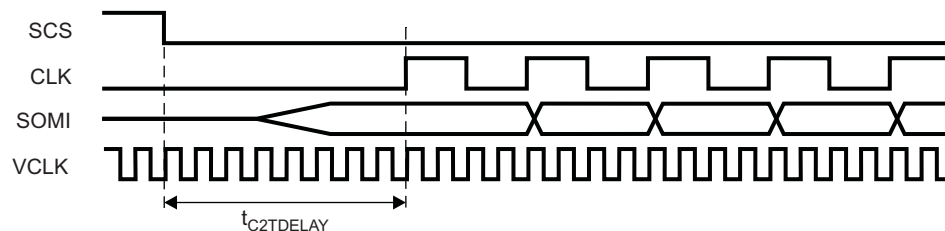
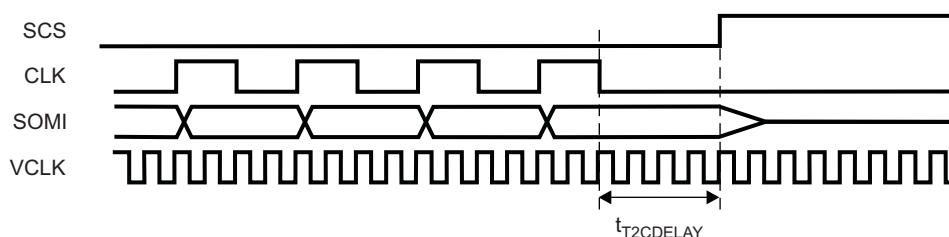
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 23-26. SPI Delay Register (SPIDELAY) Field Descriptions**

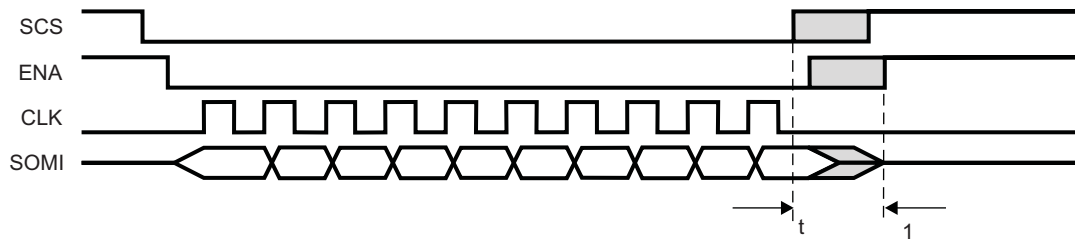
Bit	Field	Value	Description
31-24	C2TDELAY	0-1FFh	<p><b>Chip-select-active to transmit-start delay.</b> See <a href="#">Figure 23-45</a> for an example. C2TDELAY is used only in master mode. It defines a setup time (for the slave device) that delays the data transmission from the chip select active edge by a multiple of VCLK cycles.</p> <p>The setup time value is calculated as follows:  <math>t_{C2TDELAY} = (C2TDELAY + 2) * VCLK \text{ Period}</math>                      Note: If C2TDELAY = 0, then <math>t_{C2TDELAY} = 0</math>.                      Example: VCLK = 25 MHz -&gt; VCLK Period = 40ns; C2TDELAY = 07h;  <math>&gt; t_{C2TDELAY} = 360 \text{ ns}</math></p> <p>When the chip select signal becomes active, the slave has to prepare data transfer within 360 ns.</p> <p><b>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.</b></p>
23-16	T2CDELAY	0-1FFh	<p><b>Transmit-end-to-chip-select-inactive-delay.</b> See <a href="#">Figure 23-46</a> 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:  <math>t_{T2CDELAY} = (T2CDELAY + 1) * VCLK \text{ Period}</math>  <b>Note: If T2CDELAY = 0, then <math>t_{T2CDELAY} = 0</math></b>                      Example: VCLK = 25 MHz -&gt; VCLK Period = 40ns; T2CDELAY = 03h;  <math>&gt; t_{T2CDELAY} = 160 \text{ ns}</math>;</p> <p>After the last data bit (or parity bit) is being transferred the chip select signal is held active for 160 ns.</p> <p><b>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.</b></p> <p><b>Note: C2TDELAY and T2CDELAY counters do not count when SPIENA stops transmission.</b></p> <p>Both C2TDELAY and T2CDELAY counters do not have any dependency on the <math>\overline{SPIENA}</math> pin value. Even if the <math>\overline{SPIENA}</math> pin is asserted by the slave, the master will continue to delay the start of SPICLK until the C2TDELAY counter overflows.</p> <p>Similarly, even if the <math>\overline{SPIENA}</math> pin is deasserted by the slave, the master will continue to hold the SPICS pins active until the T2CDELAY counter overflows. In this way, it is guaranteed that the setup and hold times of the SPICS pins are determined by the delay timers alone. To achieve better throughput, it should be ensured that these two timers are kept at the minimum possible values.</p>

**Table 23-26. SPI Delay Register (SPIDELAY) Field Descriptions (continued)**

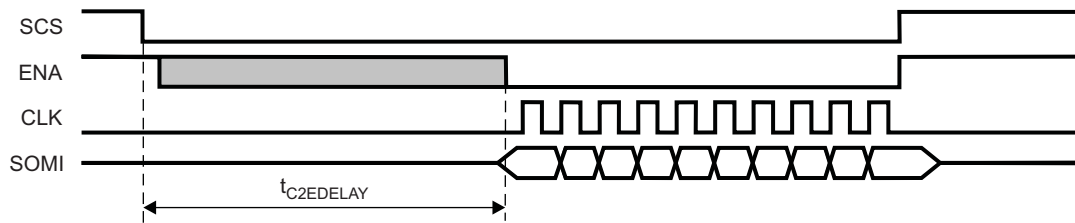
Bit	Field	Value	Description
15-8	T2EDELAY	0-1FFh	<p><b>Transmit-data-finished to ENA-pin-inactive time-out.</b> T2EDELAY is used in master mode only. It defines a time-out value as a multiple of SPI clock before SPIENA signal has to become inactive and after SPISSCS becomes inactive. SPICLK depends on 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.</p> <p>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 23-47 for an example of this condition.</p> <p><b>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:</b>  <math>t_{T2EDELAY} = T2EDELAY/SPIclock</math></p> <p>Example: SPIclock = 8 Mbit/s; T2EDELAY = 10h;  <math>&gt; t_{T2EDELAY} = 2 \mu s</math>;</p> <p>The slave device has to disable the ENA signal within 2 <math>t_s</math>, otherwise DESYNC is set and an interrupt is asserted (if enabled).</p>
7-0	C2EDELAY	0-1FFh	<p><b>Chip-select-active to ENA-signal-active time-out.</b> 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 23-48 for an example of this condition.</p> <p><b>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 an interrupt is asserted (if enabled).</b></p> <p>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.</p> <p>The timeout value is calculated as follows: <math>t_{C2EDELAY} = C2EDELAY/SPIclock</math></p> <p>Example: SPIclock = 8 Mbit/s; C2EDELAY = 30 h;  <math>&gt; t_{C2EDELAY} = 6 ms</math>;</p> <p>The slave device has to activate the ENA signal within 6 ms after the SPI has activated the chip select signal (SPISSCS), otherwise the TIMEOUT flag is set and an interrupt is asserted (if enabled).</p>

**Figure 23-45. Example:  $t_{C2DELAY} = 8$  VCLK Cycles****Figure 23-46. Example:  $t_{T2DELAY} = 4$  VCLK Cycles**

**Figure 23-47. Transmit-Data-Finished-to-ENA-Inactive-Timeout**



**Figure 23-48. Chip-Select-Active-to-ENA-Signal-Active-Timeout**



### 23.9.20 SPI Default Chip Select Register (SPIDEF)

**Figure 23-49. SPI Default Chip Select Register (SPIDEF) [offset = 4Ch]**

31	Reserved		16
	R-0		
15	8	7	0
	Reserved	CSDEF	
	R-0	R/W-1	

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

**Table 23-27. SPI Default Chip Select Register (SPIDEF) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Read returns 0. Writes have no effect.
7-0	CDEF	0-FFh	Chip select default pattern. Master-mode only.
		0	The CSDEFx bits are output to the $\overline{\text{SPISCS}}$ 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. $\overline{\text{SPISCS}}_x$ is set to 0 when no transfer is active.
		1	$\overline{\text{SPISCS}}_x$ is set to 1 when no transfer is active.

**23.9.21 SPI Data Format Registers (SPIFMT)**
**Figure 23-50. SPI Data Format Registers (SPIFMT[3:0]) [offset = 5Ch-50h]**

31	24	23	22	21	20	19	18	17	16
WDELAY		PARPOL	PARITYENA	WAITENA	SHIFTDIR	Reserved	DIS CS TIMERS	POLARITY	PHASE
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0
15	8	7	5	4	0				
PRESCALE		Reserved			CHARLEN				
R/W-0		R-0			R/W-0				

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

**Table 23-28. SPI Data Format Registers (SPIFMT[3:0]) Field Descriptions**

Bit	Field	Value	Description
31-24	WDELAY	0-3Fh	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: $WDELAY * P_{VCLK} + 2 * P_{VCLK}$ $P_{VCLK} \rightarrow$ Period of VCLK.
23	PARPOL	0 1	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	0 1	Parity enable for data format x. No parity generation/ verification is performed for this data format. 0 A parity bit is transmitted at the end of each transmitted word. At the end of a transfer the parity generator compares the received parity bit with the locally-calculated parity flag. If the parity bits do not match the RXERR flag is set in the corresponding control field. The parity type (even or odd) can be selected via the PARPOL bit. 1 <b>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.</b>
21	WAITENA	0 1	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 (C2DELAY) overflows, then the master aborts the transfer and sets the TIMEOUT error flag.
20	SHIFTDIR	0 1	<b>Shift direction for data format x.</b> 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	Reserved	0	Read returns 0. Writes have no effect.
18	DIS CS TIMERS	0 1	<b>Disable chip-select timers for this format.</b> 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.



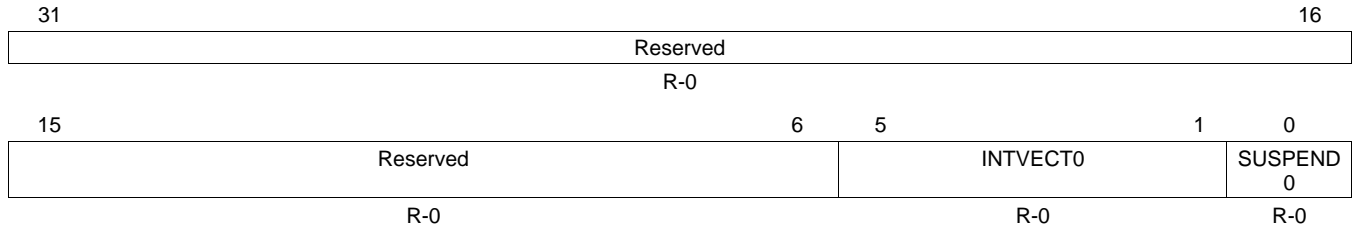
**Table 23-28. SPI Data Format Registers (SPIFMT[3:0]) Field Descriptions (continued)**

Bit	Field	Value	Description
17	POLARITY	0 1	<p><b>SPI data format x clock polarity.</b> POLARITY<sub>x</sub> defines the clock polarity of data format x.</p> <p>The following restrictions apply when switching clock phase and/or polarity:</p> <ul style="list-style-type: none"> <li>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.</li> <li>Even while using chip select pins, the polarity of SPICLK can be switched only while the slave is not selected by a valid chip select. The master SPI should ensure that while switching SPICLK polarity, it has deselected all of its slaves. Otherwise, the switching of SPICLK polarity may be incorrectly treated as a clock edge by some slaves.</li> </ul> <p>If POLARITY<sub>x</sub> is set to 0 the SPI clock signal is low-inactive, i.e., before and after data transfer the clock signal is low.</p> <p>If POLARITY<sub>x</sub> is set to 1 the SPI clock signal is high-inactive, i.e., before and after data transfer the clock signal is high.</p>
16	PHASE	0 1	<p><b>SPI data format x clock delay.</b> PHASE<sub>x</sub> defines the clock delay of data format x.</p> <p>If PHASE<sub>x</sub> 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.</p> <p>If PHASE<sub>x</sub> 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.</p>
15-8	PRESCALE		<p>SPI data format x prescaler. PRESCALE<sub>x</sub> determines the bit transfer rate of data format x if the SPI is the network master. PRESCALE<sub>x</sub> is use to derive SPICLK from VCLK. If the SPI is configured as slave, PRESCALE<sub>x</sub> <b>does not need</b> to be configured. The clock rate for data format x can be calculated as:</p> $BR_{Formatx} = VBUSPCLK / (PRESCALEx + 1)$ <p><b>Note: When PRESCALE<sub>x</sub> is set to 0, the SPI clock rate defaults to VCLK/2.</b></p>
7-5	Reserved	0	Read returns 0. Writes have no effect.
4-0	CHARLEN	0-1Fh	SPI data format x data-word length. CHARLEN <sub>x</sub> defines the word length of data format x. Legal values are 0x02 (data word length = 2 bit) to 10h (data word length = 16). Illegal values, such as 00 or 1Fh are not allowed; their effect is indeterminate.

**23.9.22 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 23-51. Interrupt Vector 0 (NTVECT0) [offset = 60h]**



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

**Table 23-29. Transfer Group Interrupt Vector 0 (INTVECT0)**

Bit	Field	Value	Description
31-6	Reserved	0	Read returns 0. Writes have no effect.
5-1	INTVECT0	0 1h + x 11h 13h 12h 14h All Other Combinations	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. <b>Note:</b> 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. There is no pending interrupt. Transfer group x (x=0,...,15) has a pending interrupt. SUSPEND0 reflects the type of interrupt ( <i>suspended</i> or <i>finished</i> ). Error Interrupt pending. The lower half of SPIFLG contains more details about the type of error. The pending interrupt is a "Receive Buffer Overrun" interrupt. <b>SPI mode:</b> The pending interrupt is a "Receive Buffer Full" interrupt. <b>Mib mode:</b> Reserved. This bit combination should not occur. <b>SPI mode:</b> The pending interrupt is a "Transmit Buffer Empty" interrupt. <b>Mib mode:</b> Reserved. This bit combination should not occur. <b>SPI mode:</b> Reserved. These bit combinations should not occur.
0	SUSPEND0	0 1	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. 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. The interrupt type is a "transfer suspended" interrupt. In other words, the transfer group referenced by INTVECT0 has asserted an interrupt because the buffer to be transferred next is in "suspend-to-wait" mode.

**NOTE:** Reading from the INTVECT0 register when "Transmit Empty" is indicated does not clear the TXINTFLG flag in the SPI Flag Register (SPIFLG) 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.

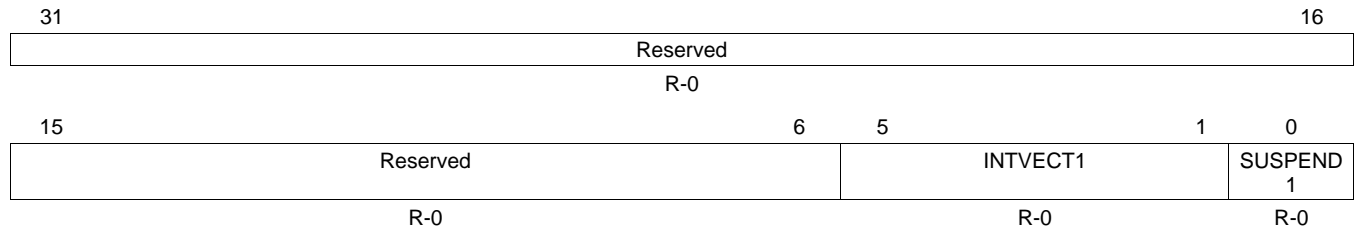
Reading the INTVECT0 register when the RXOVRN interrupt is indicated in multi-buffer mode does not clear the RXOVRN flag and hence does not clear the vector. The RXOVRN interrupt vector may be cleared in multi-buffer mode either by write-clearing the RXOVRN flag in the SPI Flag Register (SPIFLG) register or by reading the RXRAM Overrun Buffer Address Register (RXOVRN\_BUF\_ADDR) register.

---

**23.9.23 Interrupt Vector 1 (INTVECT1)**

**NOTE:** The TG interrupt is not available in SPI in compatibility mode. Therefore, there is no possibility to access this register in compatibility mode.

**Figure 23-52. Interrupt Vector 1 (INTVECT1) [offset = 64h]**



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

**Table 23-30. Transfer Group Interrupt Vector 1 (INTVECT1)**

Bit	Field	Value	Description
31-6	Reserved	0	Read returns 0. Writes have no effect.
5-1	INTVECT1	0 11h 13h 12h 14h All Other Combinations	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. <b>Note:</b> 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. There is no pending interrupt. <b>SPI mode only.</b> Error Interrupt pending. The lower half of SPIINT1 contains more details about the type of error. <b>SPI mode only.</b> The pending interrupt is a "Receive Buffer Overrun" interrupt. <b>SPI mode only.</b> The pending interrupt is a "Receive Buffer Full" interrupt. <b>SPI mode only.</b> The pending interrupt is a "Transmit Buffer Empty" interrupt. <b>SPI mode only.</b> Reserved. These bit combinations should not occur. <b>SPI mode only.</b>
0	SUSPEND1	0 1	Transfer suspended / Transfer finished interrupt flag. Every time INTVECT1 is read by the host, the corresponding interrupt flag of the referenced transfer group is cleared and INTVECT1 is updated with the vector coming next in the priority chain. 0 The interrupt type is a "transfer finished" interrupt. In other words, the buffer array referenced by INTVECT1 has asserted an interrupt because all of data from the transfer group has been transferred. 1 The interrupt type is a "transfer suspended" interrupt. In other words, the transfer group referenced by INTVECT1 has asserted an interrupt because the buffer to be transferred next is in "suspend-to-wait" mode.

**NOTE:** Reading from the INTVECT1 register when "Transmit Empty" is indicated does not clear the TXINTFLG flag in the SPI Flag Register (SPIFLG) 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 SPI Flag Register (SPIFLG) register or by reading the RXOVRN\_BUF\_ADDR register.

---

**23.9.24 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 (Bits 31:24) are used when SPIFMT3 register is selected by DFSEL[1:0] = 11 in the control field of a buffer.

**Figure 23-53. Parallel/Modulo Mode Control Register (SPIPMCTRL) [offset = 6Ch]**

31	30	29	28	26	25	24	23	22	21	20	18	17	16	
Reserved		MOD CLK POL3	MMODE 3			PMODE 3		Reserved		MOD CLK POL2	MMODE 2		PMODE 2	
R-0		R/W-0	R/W-0			R/W-0		R-0		R/W-0	R/W-0		R/W-0	
15	14	13	12	11	10	9	8	7	6	5	4	2	1	0
Reserved		MOD CLK POL1	MMODE 1			PMODE 1		Reserved		MOD CLK POL0	MMODE 0		PMODE 0	
R-0		R/W-0	R/W-0			R/W-0		R-0		R/W-0	R/W-0		R/W-0	

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

**Table 23-31. SPI Parallel/Modulo Mode Control Register (SPIPMCTRL) Field Descriptions**

Bit	Field	Value	Description
31-30	Reserved	0	Read returns 0. Writes have no effect.
29	MOD CLK POL 3	0 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.
28-26	MMODE 3	0 1h 2h 3h 4h 5h 6h-7h	These bits determine whether the SPI/MibSPI operates with 1, 2, 4, 5, or 6 data lines (if modulo option is supported by the module). 0 Normal single dataline mode - Default (PMODE should be set to 00) 1h 2-data line mode (PMODE should be set to 00) 2h 3-data line mode (PMODE should be set to 00) 3h 4-data line mode (PMODE should be set to 00) 4h 5-data line mode (PMODE should be set to 00) 5h 6-data line mode (PMODE should be set to 01) 6h-7h Reserved
25-24	PMODE 3	0 1h 2h 3h	Parallel mode bits determine whether the SPI/MibSPI operates with 1, 2, 4 or 8 data lines. 0 = normal operation/1-data line (MMODE should be set to 000) 1h = 2-data line mode (MMODE should be set to 000) 2h = 4-data line mode (MMODE should be set to 000) 3h = 8-data line mode (MMODE should be set to 000)

**Table 23-31. SPI Parallel/Modulo Mode Control Register (SPIPMCTRL) Field Descriptions (continued)**

Bit	Field	Value	Description
23-22	Reserved	0	Read returns 0. Writes have no effect.
21	MOD CLK POL 2	0	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. Normal SPICLK in all the modes.
		1	Polarity of the SPICLK will be inverted if Modulo mode is selected.
20-18	MMODE 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)
		1h	001 = 2-data line Mode (PMODE should be set to 00)
		2h	3-data line mode (PMODE should be set to 00)
		3h	4-data line mode (PMODE should be set to 00)
		4h	5-data line mode (PMODE should be set to 00)
		5h	6-data line mode (PMODE should be set to 01)
		6h-7h	Reserved
17-16	PMODE 2	0	Parallel mode bits determine whether the SPI/MibSPI operates with 1, 2, 4 or 8 data lines. 0 = normal operation/1-data line (MMODE should be set to 000)
		1h	1h = 2-data line mode (MMODE should be set to 000)
		2h	2h = 4-data line mode (MMODE should be set to 000)
		3h	3h = 8-data line mode (MMODE should be set to 000)
15-12	Reserved	0	Read returns 0. Writes have no effect.
13	MOD CLK POL 1	0	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. Normal SPICLK in all the modes.
		1	Polarity of the SPICLK will be inverted if Modulo mode is selected.
12-10	MMODE 1	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). 1-data line mode - default (PMODE should be set to 00)
		1h	2-data line mode (PMODE should be set to 00)
		2h	3-data line mode (PMODE should be set to 00)
		3h	4-data line mode (PMODE should be set to 00)
		4h	5-data line mode (PMODE should be set to 00)
		5h	6-data line mode (PMODE should be set to 01)
		6h-7h	Reserved
9-8	PMODE 1	0	Parallel mode bits determine whether the SPI/MibSPI operates with 1, 2, 4 or 8 data lines. 0 = normal operation/1-data line (MMODE should be set to 000)
		1h	1h = 2-data line mode (MMODE should be set to 000)
		2h	2h = 4-data line mode (MMODE should be set to 000)
		3h	3h = 8-data line mode (MMODE should be set to 000)
7-6	Reserved	0	Read returns 0. Writes have no effect.
5	MOD CLK POL 0	0	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. Normal SPICLK in all the modes.
		1	Polarity of the SPICLK will be inverted if Modulo mode is selected.

**Table 23-31. SPI Parallel/Modulo Mode Control Register (SPIPMCTRL) Field Descriptions (continued)**

Bit	Field	Value	Description
4-2	MMODE 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).
		0	1-data line mode - default (PMODE should be set to 00)
		1h	2-data line mode (PMODE should be set to 00)
		2h	3-data line mode (PMODE should be set to 00)
		3h	4-data line mode (PMODE should be set to 00)
		4h	5-data line mode (PMODE should be set to 00)
		5h	6-data line mode (PMODE should be set to 01)
	6h-7h	Reserved	
1-0	PMODE 0		Parallel mode bits determine whether the SPI/MibSPI operates with 1, 2, 4 or 8 data lines.
		0	0 = normal operation/1-data line (MMODE should be set to 000)
		1h	1h = 2-data line mode (MMODE should be set to 000)
		2h	2h = 4-data line mode (MMODE should be set to 000)
	3h	3h = 8-data line mode (MMODE should be set to 000)	



### 23.9.25 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 23-54. Multi-buffer Mode Enable Register (MIBSPIE) [offset = 70h]**

31	17	16
Reserved		RXRAM ACCESS
R-0		R/W-0
15	1	0
Reserved		MSPIENA
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 23-32. Multi-buffer Mode Enable Register (MIBSPIE) Field Descriptions**

Bit	Field	Value	Description
31-17	Reserved	0	Reads returns 0. Writes have no effect.
16	RXRAMACCESS	0	<b>Receive-RAM access control.</b> 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.
		1	The whole of multi-buffer RAM is fully accessible for read/write by the CPU.
			<b>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.</b>
15-1	Reserved	0	Reads returns 0. Writes have no effect.
0	MSPIENA	0	Multi-buffer mode enable. After power-up or reset, MSPIENA remains cleared, which means that the SPI runs in compatibility mode by default. If multi-buffer mode is desired, this register should be configured first after configuring the SPIGCR0 register. If MSPIENA is not set to 1, the multi-buffer mode registers are not writable.
		1	The SPI runs in compatibility mode, i.e., in this mode the MibSPI is fully code-compliant to the standard device SPI. No multi-buffered-mode features are supported.
		1	The SPI is configured to run in multi-buffer mode.

**NOTE: Accessibility of Registers**

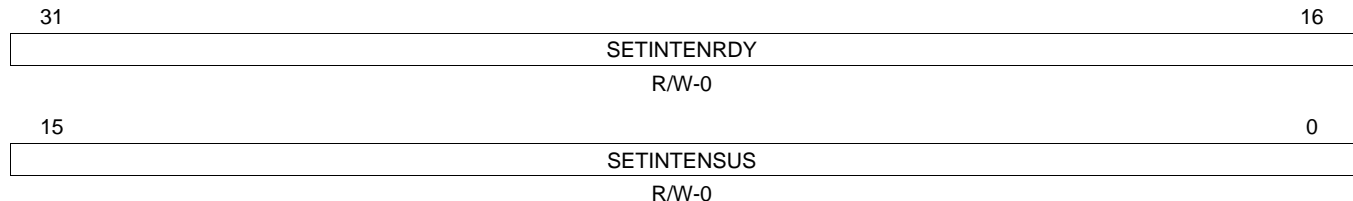
Registers from this offset address onwards are not accessible in SPI compatibility mode. They are accessible only in the multi-buffer mode.

### 23.9.26 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 23-55](#) and [Table 23-33](#) represents a super-set device with the maximum number of TGs (16) assumed. The actual number of bits available varies per device.

**Figure 23-55. TG Interrupt Enable Set Register (TGITENST) [offset = 74h]**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 23-33. TG Interrupt Enable Set Register (TGITENST) Field Descriptions**

Bit	Field	Value	Description
31-16	SET INTENRDY	0	TG interrupt set (enable) when transfer finished. <i>Read:</i> The TGx-completed interrupt is disabled. This interrupt does not get generated when TGx completes. <i>Write:</i> A write of 0 to this bit has no effect.
		1	<i>Read:</i> The TGx-completed interrupt is enabled. The interrupt gets generated when TGx completes. <i>Write:</i> Enable the TGx-completed interrupt. The interrupt gets generated when TGx completes.
15-0	SET INTENSUS	0	TG interrupt set (enabled) when transfer suspended <i>Read:</i> The TGx-completed interrupt is disabled. This interrupt does not get generated when TGx is suspended. <i>Write:</i> A write of 0 to this bit has no effect.
		1	<i>Read:</i> The TGx-completed interrupt is enabled. The interrupt gets generated when TGx is suspended. <i>Write:</i> Enable the TGx-completed interrupt. The interrupt gets generated when TGx is suspended.

### 23.9.27 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 23-56](#) and [Table 23-34](#) represents a super-set device with the maximum number of TGs (16) assumed. The actual number of bits available varies per device.

**Figure 23-56. TG Interrupt Enable Clear Register (TGITENCR) [offset = 78h]**

31	CLRINTENRDY	16
R/W-0		
15	CLRINTENSUS	0
R/W-0		

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 23-34. TG Interrupt Enable Clear Register (TGITENCR) Field Descriptions**

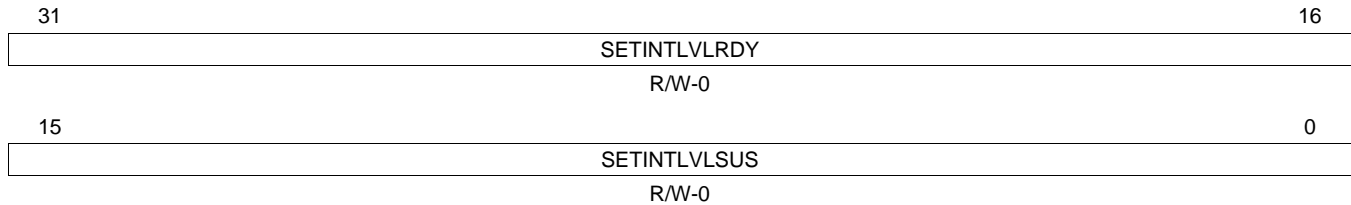
Bit	Field	Value	Description
31-16	CLR INTENRDY	0	TG interrupt clear (disabled) when transfer finished. <i>Read:</i> The TGx-completed interrupt is disabled. This interrupt does not get generated when TGx completes. <i>Write:</i> A write of 0 to this bit has no effect.
		1	<i>Read:</i> The TGx-completed interrupt is enabled. The interrupt gets generated when TGx completes. <i>Write:</i> Disable the TGx-completed interrupt. The interrupt gets generated when TGx completes.
15-0	CLR INTENSUS	0	TG interrupt clear (disabled) when transfer suspended. <i>Read:</i> The TGx-completed interrupt is disabled. This interrupt does not get generated when TGx is suspended. <i>Write:</i> A write of 0 to this bit has no effect.
		1	<i>Read:</i> The TGx-completed interrupt is enabled. The interrupt gets generated when TGx is suspended. <i>Write:</i> Disable the TGx-completed interrupt. The interrupt gets generated when TGx is suspended.

### 23.9.28 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 23-57](#) and [Table 23-35](#) represents a super-set device with the maximum number of TGs (16) assumed. The actual number of bits available varies per device.

**Figure 23-57. Transfer Group Interrupt Level Set Register (TGITLVST) [offset = 7Ch]**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 23-35. Transfer Group Interrupt Level Set Register (TGITLVST) Field Descriptions**

Bit	Field	Value	Description
31-16	SET INTLVLRDY	0	Transfer-group completed interrupt level set. <i>Read:</i> The TGx-completed interrupt is set to INT0. <i>Write:</i> A write of 0 to this bit has no effect.
		1	<i>Read:</i> The TGx-completed interrupt is set to INT1. <i>Write:</i> Set the TGx-completed interrupt to INT1.
15-0	SET INTLVLSUS	0	Transfer-group suspended interrupt level set. <i>Read:</i> TGx-suspended interrupt is set to INT0. <i>Write:</i> A write of 0 to this bit has no effect.
		1	<i>Read:</i> The TGx-suspended interrupt is set to INT1. <i>Write:</i> Set the TG-x suspended interrupt INT1.

### 23.9.29 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 23-58](#) and [Table 23-36](#) represents a super-set device with the maximum number of TGs (16) assumed. The actual number of bits available varies per device.

**Figure 23-58. Transfer Group Interrupt Level Clear Register (TGITLVCR) [offset = 80h]**

31	CLRINTLVLRDY	16
	R/W-0	
15	CLRINTLVLSUS	0
	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 23-36. Transfer Group Interrupt Level Clear Register (TGITLVCR) Field Descriptions**

Bit	Field	Value	Description
31-16	CLR INTLVLRDY	0	Transfer-group completed interrupt level clear. <i>Read:</i> The TGx-completed interrupt is set to INT0. <i>Write:</i> A write of 0 to this bit has no effect.
		1	<i>Read:</i> The TGx-completed interrupt is set to INT1. <i>Write:</i> Set the TGx-completed interrupt to INT0.
15-0	CLR INTLVLSUS	0	Transfer group suspended interrupt level clear. <i>Read:</i> TGx-suspended interrupt is set to INT0. <i>Write:</i> A write of 0 to this bit has no effect.
		1	<i>Read:</i> The TGx-suspended interrupt is set to INT1. <i>Write:</i> Set the TG-x suspended interrupt INT0.

### 23.9.30 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 23-59](#) and [Table 23-37](#) represents a super-set device with the maximum number of TGs (16) assumed. The actual number of bits available varies per device.

**Figure 23-59. Transfer Group Interrupt Flag Register (TGINTFLAG) [offset = 84h]**

31	INTFLGRDY	16
	R/W-0	
15	INTFLGSUS	0
	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

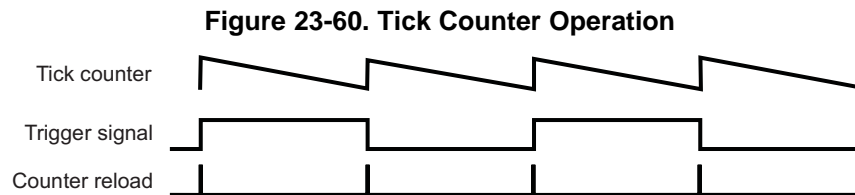
**Table 23-37. Transfer Group Interrupt Level Clear Register (TGITLVCR) Field Descriptions**

Bit	Field	Value	Description
31-16	INTFLGRDY	0	Transfer-group interrupt flag for a transfer-completed interrupt.  <b>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.</b>  <i>Read:</i> No transfer-completed interrupt occurred since last clearing of the INTFLGRDYx flag. <i>Write:</i> A write of 0 to this bit has no effect.
		1	<i>Read:</i> 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. <i>Write:</i> The corresponding bit flag is cleared.
15-0	INTFLGSUS	0	Transfer-group interrupt flag for a transfer-suspend interrupt.  <b>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.</b>  <i>Read:</i> No transfer-suspended interrupt occurred since the last clearing of the INTFLGSUSx flag. <i>Write:</i> A write of 0 to this bit has no effect.
		1	<i>Read:</i> 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. <i>Write:</i> The corresponding bit flag is cleared.

### 23.9.31 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 23-60 as a square wave, illustrates the different trigger event types for the TGs (e.g., rising edge, falling edge, and both edges).



This register is shown in Figure 23-61 and described in Table 23-38.

**Figure 23-61. Tick Count Register (TICKCNT) [offset = 90h]**

31	30	29	28	27		16
TICKENA	RELOAD	CLKCTRL				Reserved
R/W-0	R/S-0	R/W-0				R-0
15						0
TICKVALUE						
R/W-0						

LEGEND: R = Read only; R/W = Read/Write; S = Set; -n = value after reset

**Table 23-38. Tick Count Register (TICKCNT) Field Descriptions**

Bit	Field	Value	Description	
31	TICKENA	0	Tick counter enable. The internal tick counter is disabled. The counter value remains unchanged. <b>Note: When the tick counter is disabled, the trigger signal is forced low.</b>	
		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.	
30	RELOAD		<b>Pre-load the tick counter.</b> 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. <b>Note: When the tick counter is reloaded by the RELOAD bit, the trigger signal is not toggled.</b>	
29-28	CLKCTRL	0	<b>Tick counter clock source control.</b> CLKCTRL defines the clock source that is used to clock the internal tick counter. SPICLK of data word format 0 is selected as the clock source of the tick counter	
		1h		SPICLK of data word format 1 is selected as the clock source of the tick counter
		1h		SPICLK of data word format 2 is selected as the clock source of the tick counter
		3h		SPICLK of data word format 3 is selected as the clock source of the tick counter
27-16	Reserved	0	Read returns 0. Writes have no effect.	
15-0	TICKVALUE	0-FFFFh	<b>Initial value for the tick counter.</b> 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.	

**23.9.32 Last TG End Pointer (LTGPEND)**
**Figure 23-62. Last TG End Pointer (LTGPEND) [offset = 94h]**

31	29	28	24	23	16
Reserved		TG IN SERVICE			Reserved
R-0		R-0			R-0
15	14			8	7
Rsvd.	LPEND				Reserved
R-0	R/W-0				R-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 23-39. Last TG End Pointer (LTGPEND) Field Descriptions**

Bit	Field	Value	Description
31-29	Reserved	0	Reads returns 0. Writes have no effect.
28-24	TG IN SERVICE	0 1h ... 10h 11h-1Fh	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. No TG is being serviced by the sequencer. TG0 is being serviced by the sequencer. ... TG15 is being serviced by the sequencer. <b>Note: The number of transfer groups varies by device.</b> Invalid values
23-15	Reserved	0	Reads returns 0. Writes have no effect.
14-8	LPEND	0-7Fh	<b>Last TG end pointer.</b> 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. <b>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.</b> <b>Note: The number of transfer groups varies by device, thus the last TG also varies by device.</b>
7-0	Reserved	0	Reads returns 0. Writes have no effect.



### 23.9.33 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 23-63. MibSPI TG Control Registers (TGxCTRL) [offsets = 98h-D4h]**

31	30	29	28	27	24	23	20	19	16
TGEN A	ONES HOT	PRST	TGTD	Reserved			TRIGEVT	TRIGSRC	
R/W-0	R/W-0	R/W-0	R-0	R-0			R/W-0	R/W-0	
15	14				8	7	6	0	
Rsvd.	PSTART				Rsvd.	PCURRENT			
R-0	R/W-0				R-0	R-0			

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 23-40. TG Control Registers (TGxCTRL) Field Descriptions**

Bit	Field	Value	Description
31	TGENA	0 1	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	ONESH OTx	0 1	Single transfer for TGx. 0 TGx initiates a transfer every time a trigger event occurs and TGENA is set. 1 A transfer from TGx will be performed only once (one shot) after a valid trigger event at the selected trigger source. After the transfer is finished the TGENAx control bit will be cleared and therefore no additional transfer can be triggered before the host enables the TG again. This one shot mode ensures that after one group transfer the host has enough time to read the received data and to provide new transmit data.
29	PRSTx	0 1	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. <b>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.</b> 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	0 1	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	0	Read returns 0. Writes have no effect.

**Table 23-40. TG Control Registers (TGxCTRL) Field Descriptions (continued)**

Bit	Field	Value	Description
23-20	TRIGEV Tx		Type of trigger event. A level-triggered TG can be stopped by de-activating the level trigger. However, the following restrictions apply. <ul style="list-style-type: none"> <li>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.</li> <li>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.</li> <li>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.</li> </ul>
		0	never Never trigger TGx. This is the default value after reset.
		1h	rising edge A rising edge (0 to 1) at the selected trigger source (TRIGSRCx) initiates a transfer for TGx
		2h	falling edge A falling edge (1 to 0) at the selected trigger source (TRIGSRCx) initiates a transfer for TGx
		3h	both edges Rising and falling edges at the selected trigger source (TRIGSRCx) initiates a transfer for TGx
		4h	Reserved
		5h	high-active While the selected trigger source (TRIGSRCx) is at a logic high level (1) the group transfer is continued and at the end of one group transfer restarted at the beginning. If the logic level changes to low (0) during an ongoing group transfer, the whole group transfer will be stopped. <b>Note: If ONESHOTx is set the transfer is performed only once.</b>
		6h	low-active While the selected trigger source (TRIGSRCx) is at a logic low level (0) the group transfer is continued and at the end of one restarted at the beginning. If the logic level changes to high (1) during an ongoing group transfer, the whole group transfer will be stopped. <b>Note: If ONESHOTx is set the transfer is performed only once.</b>
		7h	always A repetitive group transfer will be performed. <b>Note: By setting the TRIGSRC to 0, the TRIGEV to 7h (ALWAYS), and the ONESHOTx bit to 1, software can trigger this TG. Upon setting the TGENA bit, the TG is immediately triggered.</b> <b>Note: If ONESHOTx is set the transfer is performed only once.</b>
		1xxx	Reserved

**Table 23-40. TG Control Registers (TGxCTRL) Field Descriptions (continued)**

Bit	Field	Value	Description
19-16	TRIGSR Cx	0 1h 2h 3h 4h 5h 6h 7h 8h 9h Ah Bh Ch Dh Eh Fh	<b>Trigger source.</b> After reset, the trigger sources of all TGs are disabled. disabled EXT0 External trigger source 0. The actual source varies per device (e.g. HET I/O channel, event pin, etc.). EXT1 External trigger source 1. The actual source varies per device (e.g. HET I/O channel, event pin, etc.). EXT2 External trigger source 2. The actual source varies per device (e.g. HET I/O channel, event pin, etc.). EXT3 External trigger source 3. The actual source varies per device (e.g. HET I/O channel, event pin, etc.). EXT4 External trigger source 4. The actual source varies per device (e.g. HET I/O channel, event pin, etc.). EXT5 External trigger source 5. The actual source varies per device (e.g. HET I/O channel, event pin, etc.). EXT6 External trigger source 6. The actual source varies per device (e.g. HET I/O channel, event pin, etc.). EXT7 External trigger source 7. The actual source varies per device (e.g. HET I/O channel, event pin, etc.). EXT8 External trigger source 8. The actual source varies per device (e.g. HET I/O channel, event pin, etc.). EXT9 External trigger source 9. The actual source varies per device (e.g. HET I/O channel, event pin, etc.). EXT10 External trigger source 10. The actual source varies per device (e.g. HET I/O channel, event pin, etc.). EXT11 External trigger source 11. The actual source varies per device (e.g. HET I/O channel, event pin, etc.). EXT12 External trigger source 12. The actual source varies per device (e.g. HET I/O channel, event pin, etc.). EXT13 External trigger source 13. The actual source varies per device (e.g. HET I/O channel, event pin, etc.). TICK Internal periodic event trigger. The tick counter can initiate periodic group transfers.
15	Reserved	0	Read returns 0. Writes have no effect.
14-8	PSTARTx	0-7Fh	<b>TG start address.</b> 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: <ul style="list-style-type: none"> <li>• The TG is enabled.</li> <li>• The end of the TG is reached during a transfer.</li> <li>• A trigger event occurs while PRST is set to 1.</li> </ul>
7	Reserved	0	Read returns 0. Writes have no effect.
6-0	PCURRENTx	0-7Fh	Pointer to current buffer. PCURRENT is read-only. PCURRENTx stores the address (0...127) of the buffer that corresponds to this TG. If the TG switches from active transfer mode to suspend to wait, 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.

### 23.9.34 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 23-64. DMA Channel Control Register (DMAxCTRL) [offset = D8h-F4h]**

31	30							24	23				20	19				16
ONES HOT		BUFID						RXDMA_MAP			TXDMA_MAP							
R/W-0		R/W-0						R/W-0			R/W-0							
15	14	13	12				8	7	6	5					0			
RXDM AENA	TXDM AENA	NOBR K	ICOUNT			Rsvd.	COUN T BIT17	COUNT										
R/W-0	R/W-0	R/W-0	R/W-0			R-0	R-0	R-0										

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 23-41. DMA Channel Control Register (DMAxCTRL) Field Descriptions**

Bit	Field	Value	Description
31	ONESHOT	0  1	Auto-disable of DMA channel after ICOUNT+1 transfers.  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.  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_MA Px	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.
19-16	TXDMA_MA Px	0-Fh	Each MibSPI DMA channel can be linked to two physical DMA Request lines of the DMA controller. One request line for receive data and the other for request line for transmit data.  TXDMA_MAPx[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	0  1	Receive data DMA channel enable.  0 No DMA request upon new receive data.  1 The physical DMA channel for the receive path is enabled. The first DMA request pulse is generated after the first transfer from the referenced buffer (BUFIDx) is finished. The buffer should be configured in as "skip until RXEMPTY is set" or "suspend to wait until RXEMPTY is set" in order to ensure synchronization between the DMA controller and the MibSPI sequencer.

**Table 23-41. DMA Channel Control Register (DMAxCTRL) Field Descriptions (continued)**

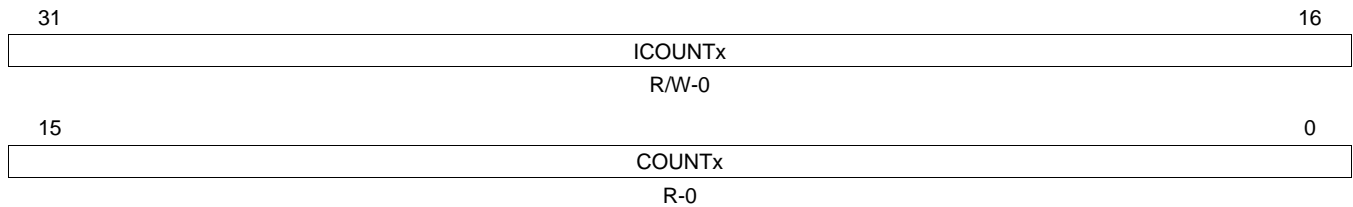
Bit	Field	Value	Description
14	TXDAMENAx	0 1	<p>Transmit data DMA channel enable.</p> <p>0 No DMA request upon new transmit data.</p> <p>1 The physical DMA channel for the transmit path is enabled. The first DMA request pulse is generated right after setting TXDMAENAx to load the first transmit data. 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.</p>
13	NOBRKx	0 1	<p>Non-interleaved DMA block transfer. This bit is available in master mode only.</p> <p><b>Note: Special Conditions during a NOBRK Buffer Transfer.</b> If a NOBRK DMA buffer is currently being serviced by the sequencer, then it is not allowed to be disabled prematurely.</p> <p><b>During a NOBRK transfer, the following operations are not allowed:</b></p> <ul style="list-style-type: none"> <li>• Clearing the NOBRKx bit to 0</li> <li>• Clearing the RXDMAENAx to 0 (if it is already 1)</li> <li>• Clearing the TXDMAENAx to 0 (if it is already 1)</li> <li>• Clearing the BUFMODE[2:0] bits to 000</li> </ul> <p><b>Any attempts to perform these actions during a NOBRK transfer will produce unpredictable results.</b></p> <p>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.</p> <p>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.</p> <p><b>Note: Triggering of higher priority TGs or enabling of higher priority DMA channels will not interrupt a NOBRK block transfer.</b></p>
12-8	ICOUNTx	0-1Fh	<p><b>Initial count of DMA transfers.</b> 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.</p> <p>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.</p> <p><b>Note: See Section 23.9.35 (ICOUNT) and Section 23.9.36 (DMACNTLEN) about how to increase the ICOUNT to a 16-bit value. With this extended capability, MibSPI can transfer a block of up to 65535 (65K) words without interleaving (if NOBRK is used) or without deasserting the chipselect between the buffers (if CSHOLD is used).</b></p>
7	Reserved	0	Read returns 0. Writes have no effect.
6	COUNT BIT17x		The 17th bit of the COUNT field of DMAxCOUNT register.
5-0	COUNTx	0-3Fh	<p>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.</p> <p><b>Note: If the TX and RX DMA requests are enabled, the COUNT register will be decremented when the RX has been serviced.</b></p>

**23.9.35 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 23-65. DMAxCOUNT Register (ICOUNT) [offset = F8h-114h]**



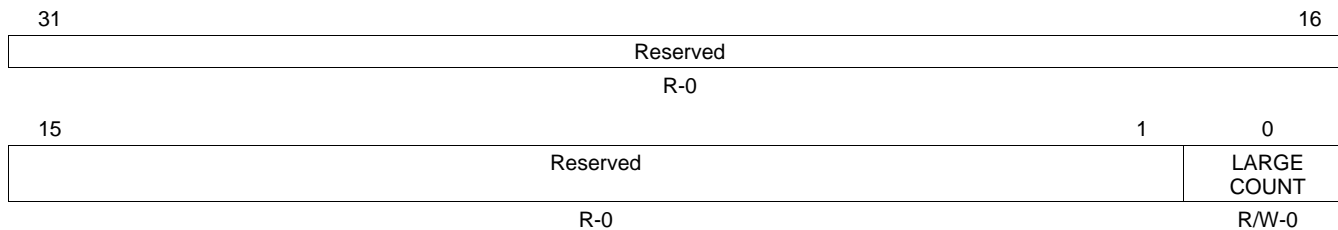
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 23-42. MibSPI DMAxCOUNT Register (ICOUNT) Field Descriptions**

Bit	Field	Value	Description
31-16	ICOUNTx	0-FFFFh	<p><b>Initial number of DMA transfers.</b> 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</p>
15-0	COUNTx	0-FFFFh	<p><b>The actual number of remaining DMA transfers.</b> 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.</p> <p><b>Note: Usage Tip for Block Transfer Using a Single DMA Request.</b> 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.</p>

### 23.9.36 DMA Large Count (DMACNTLEN)

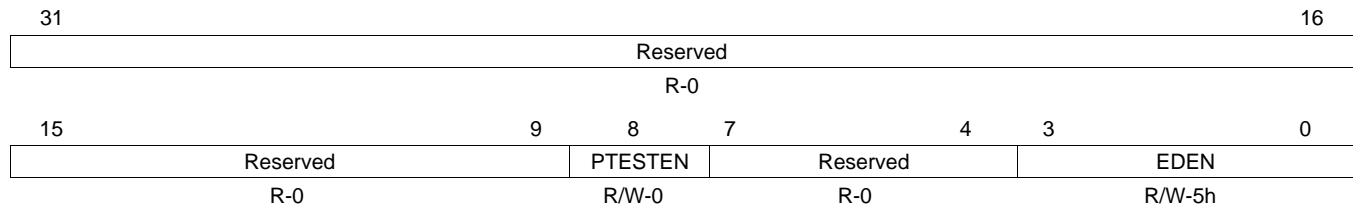
**Figure 23-66. DMA Large Count Register (DMACNTLEN) [offset = 118h]**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 23-43. MibSPI DMA Large Count Register (DMACNTLEN) Field Descriptions**

Bit	Field	Value	Description
31-1	Reserved	0	Read returns 0. Writes have no effect.
0	LARGE COUNT	0	Select either the 16-bit DMAxCOUNT counters or the smaller counters in DMAxCTRL. <b>Select the DMAxCTRL counters.</b> 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	<b>Select the DMAxCOUNT counters.</b> 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.

**23.9.37 Multi-buffer RAM Uncorrectable Parity Error Control Register (UERRCTRL)**
**Figure 23-67. Multi-buffer RAM Uncorrectable Parity Error Control Register (UERRCTRL) [offset = 120h]**


LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 23-44. Multi-buffer RAM Uncorrectable Parity Error Control Register (UERRCTRL) Field Descriptions**

Bit	Field	Value	Description
31-9	Reserved	0	Reads returns 0. Writes have no effect.
8	PTESTEN	0	<b>Parity memory test enable.</b> 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 <a href="#">Section 23.11</a> for further details about parity memory testing. Parity bits are not memory-mapped.
		1	
7-4	Reserved	0	Reads returns 0. Writes have no effect.
3-0	EDEN	5h	Error detection enable. These bits enable parity error detection. Parity error detection logic (default) is disabled.
		All Other Values	
			<b>Note: It is recommended to write a 1010 to enable error detection, to guard against a soft error from disabling parity error detect</b>



### 23.9.38 Multi-buffer RAM Uncorrectable Parity Error Status Register (UERRSTAT)

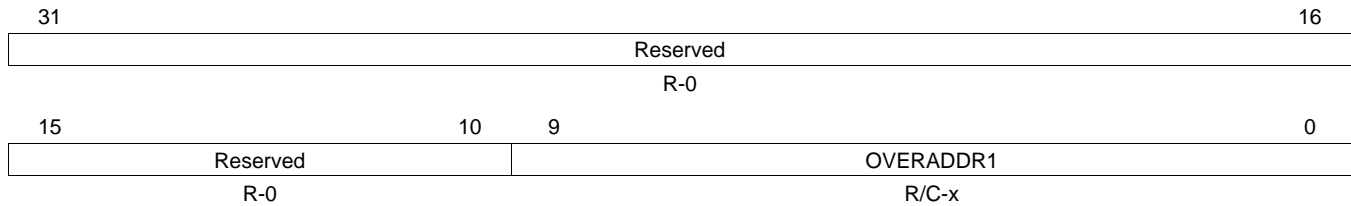
**Figure 23-68. Multi-buffer RAM Uncorrectable Parity Error Status Register (UERRSTAT) [offset = 124h]**

31	Reserved			16	
R-0					
15	Reserved		2	1	0
R-0			EDFLG1	EDFLG0	
			R/C-0	R/C-0	

LEGEND: R/W = Read/Write; R = Read only; C = Clear; -n = value after reset

**Table 23-45. Multi-buffer RAM Uncorrectable Parity Error Status Register (UERRSTAT) Field Descriptions**

Bit	Field	Value	Description
31-2	Reserved	0	Reads returns 0. Writes have no effect.
1	EDFLG1	0	Uncorrectable parity error detection flag. This flag indicates if a parity error occurred in the RXRAM. <b>Note: Reading the UERRADDR1 register clears the EDFLG1 bit.</b> <i>Read:</i> No error has occurred. <i>Write:</i> Writing a zero to this bit has no effect.
		1	<i>Read:</i> An error was detected and the address is captured in the UERRADDR1 register. <i>Write:</i> The bit is cleared to 0.
0	EDFLG0	0	Uncorrectable parity error detection flag. This flag indicates if a parity error occurred in the TXRAM. <b>Note: Reading the UERRADDR0 register clears the EDFLG0 bit.</b> <i>Read:</i> No error has occurred. <i>Write:</i> Writing a zero to this bit has no effect.
		1	<i>Read:</i> An error was detected and the address is captured in the UERRADDR0 register. <i>Write:</i> The bit is cleared to 0.

**23.9.39 RXRAM Uncorrectable Parity Error Address Register (UERRADDR1)**
**Figure 23-69. RXRAM Uncorrectable Parity Error Address Register (UERRADDR1) [offset = 128h]**


LEGEND: R/W = Read/Write; R = Read only; C = Clear; -n = value after reset

**Table 23-46. RXRAM Uncorrectable Parity Error Address Register (UERRADDR1) Field Descriptions**

Bit	Field	Value	Description
31-10	Reserved	0	Read returns 0. Writes have no effect.
9-0	OVERADDR1	200-3FFh	<p><b>Uncorrectable parity error address for RXRAM.</b> This register holds the address where a parity error is generated while reading RXRAM. Only the CPU or DMA can read from RXRAM locations. The address captured is byte-aligned. This error address is frozen from being updated until it is read by the CPU. The offset address of RXRAM varies from 200h - 3FFh.</p> <p>The register does not clear its contents during or after module-level reset, system-level reset or even power-up reset.</p> <p>A read operation to this register clears its contents to the default value 200h.</p> <p>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.</p> <p><b>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.</b></p>

### 23.9.40 TXRAM Uncorrectable Parity Error Address Register (UERRADDR0)

Figure 23-70. TXRAM Uncorrectable Parity Error Address Register (UERRADDR0) [offset = 12Ch]

31	Reserved			16
R-0				
15	9	8	0	
Reserved			UERRADDR0	
R-0			R/C-x	

LEGEND: R/W = Read/Write; R = Read only; C = Clear; -n = value after reset

Table 23-47. TXRAM Uncorrectable Parity Error Address Register (UERRADDR0) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Read returns 0. Writes have no effect.
8-0	UERRADDR1	0-1FFh	<p><b>Uncorrectable parity error address for TXRAM.</b> This register holds the address where a parity error is generated while reading from TXRAM. The TXRAM can be read either by CPU or by the MibSPI sequencer logic for transmission. The address captured is byte-aligned. This error address is frozen from being updated until it is read by the CPU. The offset address of TXRAM varies from 0 - 1FFh.</p> <p>The register does not clear its contents during or after module-level reset, system-level reset, or even power-up reset.</p> <p>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.</p> <p><b>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.</b></p>

**23.9.41 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 23-71. RXRAM Overrun Buffer Address Register (RXOVRN\_BUF\_ADDR) [offset = 130h]**

31	Reserved	16
R-0		
15	10      9	0
Reserved	RXOVRN_BUF_ADDR	
R-0	R-200h	

LEGEND: R/W = Read/Write; R = Read only; C = Clear; -n = value after reset

**Table 23-48. RXRAM Overrun Buffer Address Register (RXOVRN\_BUF\_ADDR) Field Descriptions**

Bit	Field	Value	Description
31-10	Reserved	0	Read returns 0. Writes have no effect.
9-0	RXOVRN_BUF_ADDR	200-3FCh	<p><b>Address in RXRAM at which an overwrite occurred.</b> 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.</p> <p>This word-aligned address can vary from 0x200 - 0x3FC. Contents of this register are valid only when any of the INTVECT0 or INTVECT1 and SPIFLG registers show an RXOVRN error vector while in multi-buffer mode. If there are multiple overrun errors, then this register holds the address of first overrun address until it is read.</p> <p><b>Note: Reading this register clears the RXOVRN interrupt flag in the SPIFLG register and the TGINTVECTx.</b></p> <p><b>Note: Receiver overrun errors in multi-buffer mode can be completely avoided by using the <i>SUSPEND until RXEMPTY</i> feature, which can be programmed into each buffer of any TG. However, using the <i>SUSPEND until RXEMPTY</i> 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.</b></p>

### 23.9.42 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 5h.

**Figure 23-72. I/O-Loopback Test Control Register (IOLPBKTSTCR) [offset = 134h]**

31		25	24	23		21	20	19	18	17	16		
Reserved				SCS FAIL FLG	Reserved			CTRL BIT ERR	CTRL DES YNC	CTRL PAR ERR	CTRL TIME OUT	CTRL DLEN ERR	
R-0				R/C-0	R-0			R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
15		12	11		8	7	6	5		3	2	1	0
Reserved		IOLPBKTSTENA			Reserved			ERR SCS PIN		CTRL SCSPI NERR	LPBKT YPE	RXPE NA	
R-0		R/W-5h			R-0			R/W-0		R/W-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; C = Clear; -n = value after reset

**Table 23-49. I/O-Loopback Test Control Register (IOLPBKTSTCR) Field Descriptions**

Bit	Field	Value	Description
31-25	Reserved	0	Read returns 0. Writes have no effect.
24	SCS FAIL FLG	0	Bit indicating a failure on $\overline{\text{SPISCS}}$ pin compare during analog loopback. <i>Read:</i> No mismatches occurred on any of the eight chip select pins (vs. the internal chip select number CSNR during transfers). <i>Write:</i> Writing a zero to this bit has no effect.
		1	<i>Read:</i> 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. <i>Write:</i> This flag bit is cleared.
23-21	Reserved	0	Read returns 0. Writes have no effect.
20	CTRL BITERR	0	Controls inducing of BITERR during I/O loopback test mode. Do not interfere with looped-back data.
		1	Induces bit errors by inverting the value of the incoming data during loopback.
19	CTRL DESYNC	0	Controls inducing of the desync error during I/O loopback test mode. 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	0	Controls inducing of the parity errors during I/O loopback test mode. Do not cause a parity error.
		1	Induce a parity error by inverting the polarity of the parity bit.
17	CTRL TIMEOUT	0	Controls inducing of the timeout error during I/O loopback test mode. 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	0	Controls inducing of the data length error during I/O loopback test mode. Do not cause a data-length error.
		1	Induce a data-length error. <i>Master mode:</i> The $\overline{\text{SPIENA}}$ pin (if functional) is forced to 1 when the module starts shifting data. <i>Slave mode:</i> The incoming $\overline{\text{SPISCS}}$ pin (if functional) is forced to 1 when the module starts shifting data.
15-12	Reserved	0	Read returns 0. Writes have no effect.

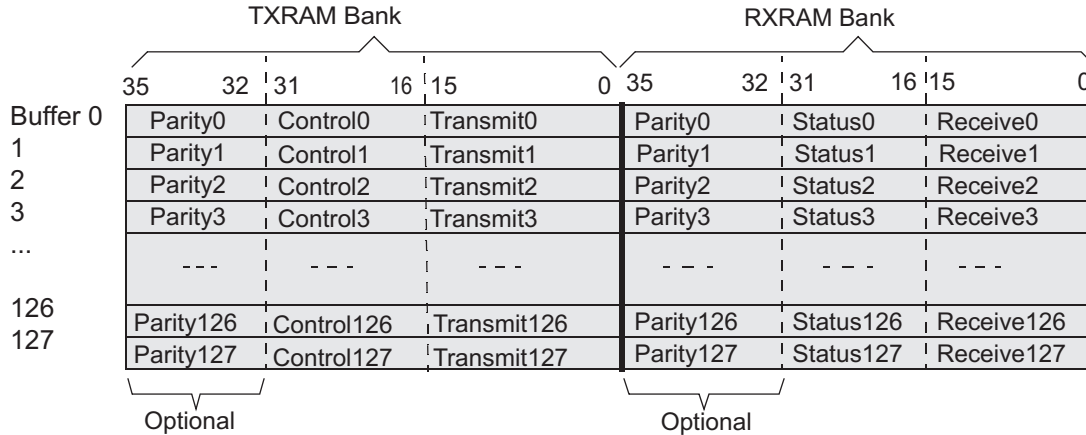
**Table 23-49. I/O-Loopback Test Control Register (IOLPBKTSTCR) Field Descriptions (continued)**

Bit	Field	Value	Description
11-8	IOLPBKSTENA	Ah All Other Values	Module I/O loopback test enable key. Enable I/O loopback test mode. Disable I/O loopback test mode.
7-6	Reserved	0	Read returns 0. Writes have no effect.
5-3	ERR SCS PIN	0 1h ... 7h	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. Select $\overline{\text{SPISCS}}[0]$ for injecting error Select $\overline{\text{SPISCS}}[1]$ for injecting error ... Select $\overline{\text{SPISCS}}[7]$ for injecting error.
2	CTRL SCS PIN ERR	0 1	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. Disable the $\overline{\text{SPISCS}}[3:0]$ error-inducing logic. Enable the $\overline{\text{SPISCS}}[3:0]$ error-inducing logic.
1	LPBK TYPE	0 1	<b>Module I/O loopback type (analog/digital).</b> See <a href="#">Figure 23-22</a> for the different types of loopback modes. Enable Digital loopback when IOLPBKTSTENA = 1010. Enable Analog loopback when IOLPBKTSTENA = 1010.
0	RXP ENA	0 1	Enable analog loopback through the receive pin. <b>Note: This bit is valid only when LPBK TYPE = 1, which chooses analog loopback mode.</b> Analog loopback is through the transmit pin. Analog loopback is through the receive pin.

### 23.10 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 23-73. A 4-bit parity field per word is also included in each bank of RAM.

**Figure 23-73. 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.

**NOTE:** Write to unimplemented buffer is overwriting the corresponding implemented buffer. In MIBSPI, if the RAM SIZE specified is 32 buffers, write to 33rd buffer overwrites 1st buffer, write to 34th buffer overwrites 2st buffer and so on.

#### 23.10.1 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 MSINENA System module register. This bit is device-specific for each memory that support auto-initialization. Please refer to the device datasheet to identify the control bit for the multi-buffer RAM. This starts the initialization process. The BUFINITACTIVE bit will get set to reflect that the initialization is ongoing.
3. When the memory initialization is completed, the corresponding status bit in the MINISTAT register will be set. Also, the BUFINITACTIVE bit will get cleared.
4. Disable the global hardware memory initialization key by programming a value of 0101 to the bits [3:0] of the MINITGCR register of the System module.

Please refer to the Architecture User Guide for more details on the memory auto-initialization process.

---

**NOTE:** During Auto Initialization process, all the Multibuffer mode registers (except MIBSPIE) will be reset to their default values. So, it should be ensured that Auto Initialization is completed before configuring the Multibuffer mode registerse.

---



### 23.10.2 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 FF0E 0000h.

**Table 23-50. Multi-buffer RAM Register Summary**

Offset	Acronym	Register Description	Section
Base + 000-1FFh	TXRAM	Multi-buffer RAM Transmit Data Register	<a href="#">Section 23.10.3</a>
Base + 200-3FFh	RXRAM	Multi-buffer RAM Receive Buffer Register	<a href="#">Section 23.10.4</a>

### 23.10.3 Multi-buffer RAM Transmit Data Register

Each word of TXRAM is a transmit-buffer register.

**Figure 23-74. Multi-buffer RAM Transmit Data Register [offset = Base + 000-1FFh]**

31	29	28	27	26	25	24	23	16
BUFMODE		CSHOLD	LOCK	WDEL	DFSEL		CSNR	
R/W-0		R/W-0	R/W-0	R/W-0	R/W-0		R/W-0	
15								0
TXDATA								
R/W-0								

LEGEND: R = Read only; R/W = Read/Write; -n = value after reset

**Table 23-51. Multi-buffer RAM Transmit Data Register Field Descriptions**

Bit	Field	Value	Description
31-29	BUFMODE		<p>Specify conditions that are recognized by the sequencer to initiate transfers of each buffer word.</p> <p>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 data transfer.</p> <p>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.</p> <p>0 <b>disabled.</b> The buffer is disabled</p> <p>1h <b>skip single-transfer mode.</b> Skip this buffer until the corresponding TXFULL flag is set (i.e. new transmit data is available).</p> <p>2h <b>skip overwrite-protect mode.</b> Skip this buffer until the corresponding RXEMPTY flag is set (i.e. new receive data can be stored in RXDATA without data loss).</p> <p>3h <b>skip single-transfer overwrite-protect mode.</b> 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).</p> <p>4h <b>continuous mode.</b> 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.</p> <p>5h <b>suspend single-transfer mode.</b> 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).</p> <p>6h <b>suspend overwrite-protect mode.</b> 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).</p> <p>7h <b>suspend single-transfer overwrite-protect mode.</b> 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).</p>
28	CSHOLD		<p>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.</p> <p>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.</p> <p>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.</p>
27	LOCK		<p>Lock two consecutive buffer words. Do not allow interruption by TG's with higher priority.</p> <p>0 Any higher-priority TG can begin at the end of the current transaction.</p> <p>1 A higher-priority TG cannot occur until after the next unlocked buffer word is transferred.</p>

**Table 23-51. Multi-buffer RAM Transmit Data Register Field Descriptions (continued)**

Bit	Field	Value	Description
26	WDEL	0  1	<p>Enable the delay counter at the end of the current transaction.</p> <p>Note: The WDEL bit is supported in master mode only. In slave mode, this bit will be ignored.</p> <p>0 No delay will be inserted. However, SPISCS pins will still be de-activated for at least for 2VCLK cycles if CSHOLD = 0.</p> <p>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).</p> <p>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 SPISCS pins will be de-activated for at least (WDELAY + 2) * VCLK_Period duration.</p>
25-24	DFSEL	0 1h 2h 3h	<p>Data word format select</p> <p>0 Data word format 0 is selected</p> <p>1h Data word format 1 is selected</p> <p>2h Data word format 2 is selected</p> <p>3h Data word format 3 is selected</p>
32-16	CSNR	0-FFh	<p>Chip select number. CSNR defines the chip-select that will be activated during the data transfer.</p> <p><b>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.</b></p> <p><b>Note: 2) Bits [23:20] is not writable in the device due to non availability of Chip select pins CS[4:7].</b></p>
15-0	TXDATA	0-7FFFh	<p><b>Transfer data.</b> 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.</p> <p>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 0.</p> <p>Write to this register ONLY when using the automatic slave chip-select feature (see <a href="#">Section 23.2</a> 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.</p> <p>When this register is read, the contents of TXBUF, which holds the latest data written, will be returned.</p> <p><b>Note: Regardless of the character length, the transmit data should be right-justified before writing to the SPIDAT1 register.</b></p>

### 23.10.4 Multi-buffer RAM Receive Buffer Register

Each word of RXRAM is a receive-buffer register.

**Figure 23-75. Multi-buffer RAM Receive Buffer Register [offset = RAM Base + 200-3FFh]**

31	30	29	28	27	26	25	24	23	16
RXEM PTY	RXOV R	TXFUL L	BITER R	DESY NC	PARIT YERR	TIMEO UT	DLEN ERR	LCSNR	
RS-1	RC-0	R-0	RC-0	RC-0	RC-0	RC-0	RC-0	R-0	
15								0	
RXDATA									
R/W-0									

LEGEND: R = Read only; R/W = Read/Write; C = Clear; S = Set; -n = value after reset

**Table 23-52. Multi-buffer Receive Buffer Register Field Descriptions**

Bit	Field	Value	Description
31	RXEMPTY	0 1	<p><b>Receive data buffer empty.</b> 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.</p> <p>New data has been received and copied into the SPIBUF field.</p> <p>No data has been received since the last read of SPIBUF.</p> <p>This flag gets set to 1 under the following conditions:</p> <ul style="list-style-type: none"> <li>Reading the RXDATA portion of the SPIBUF register.</li> <li>Writing a 1 to clear the RXINTFLG bit in the SPI Flag Register (SPIFLG) register.</li> </ul> <p>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).</p>
30	RXOVR	0 1	<p>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).</p> <p>If enabled, the RXOVRN interrupt is generated when RXBUF is overwritten, and reading either SPI Flag Register (SPIFLG) or SPIVECTx shows the RXOVRN condition. Two read operations from the SPIBUF register are required to reach the overwritten buffer word (one to read SPIBUF, which then transfers RXDATA into SPIBUF for the second read).</p> <p>This flag is cleared to 0 when the RXDATA is read.</p> <p><b>Note: A special condition under which RXOVR flag gets set. If both SPIBUF and RXBUF are already full and while another buffer receive is underway, if any errors such as TIMEOUT, BITERR and DLEN_ERR occur, then RXOVR in RXBUF and SPI Flag Register (SPIFLG) registers will be set to indicate that the status flags are getting overwritten by the new transfer. This overrun should be treated like a normal receive overrun.</b></p> <p>0 No receive data overrun condition occurred since last read of the data field.</p> <p>1 A receive data overrun condition occurred since last read of the data field.</p>
29	TXFULL	0 1	<p><b>Transmit data buffer full.</b> 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.</p> <p>0 The transmit buffer is empty; SPIDAT0/SPIDAT1 is ready to accept a new data.</p> <p>1 The transmit buffer is full; SPIDAT0/SPIDAT1 is not ready to accept new data.</p>
28	BITERR	0 1	<p><b>Bit error.</b> There was a mismatch of internal transmit data and transmitted data.</p> <p>0 No bit error occurred.</p> <p><b>Note: This flag is cleared to 0 when the RXDATA portion of the SPIBUF register is read.</b></p> <p>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.</p>

**Table 23-52. Multi-buffer Receive Buffer Register Field Descriptions (continued)**

Bit	Field	Value	Description
27	DESYNC	0 1	<p><b>Desynchronization of slave device.</b>This bit is valid in master mode only.</p> <p>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 <math>t_{2EDELAY}</math>. If DESYNCENA is set, an interrupt is asserted. Desynchronization can occur if a slave device misses a clock edge coming from the master.</p> <p><b>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.</b></p> <p>0 No slave desynchronization detected.</p> <p><b>Note: This flag is cleared to 0 when the RXDATA portion of the SPIBUF register is read.</b></p> <p>1 A slave device is desynchronized.</p>
26	PARITYERR	0 1	<p><b>Parity error.</b>The calculated parity differs from the received parity bit.</p> <p>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.</p> <p><b>Note: This flag is cleared to 0 when the RXDATA portion of the SPIBUF register is read.</b></p> <p>0 No parity error detected.</p> <p>1 A parity error occurred.</p>
25	TIMEOUT	0 1	<p>Time-out because of non-activation of ENA pin.</p> <p>The SPI generates a time-out when the slave does not respond in time by activating the ENA signal after the chip select signal has been activated. If a time-out condition is detected, the corresponding chip select is deactivated immediately and the TIMEOUT flag is set. In addition, the TIMEOUT flag in the status field of the corresponding buffer and in the SPI Flag Register (SPIFLG) register is set.</p> <p><b>This bit is valid only in master mode.</b></p> <p>This flag is cleared to 0 when RXDATA portion of the SPIBUF register is read.</p> <p>0 No ENA-pin time-out occurred.</p> <p>1 An ENA signal time-out occurred.</p>
24	DLENERR	0 1	<p><b>Data length error flag.</b></p> <p><b>Note: This flag is cleared to 0 when the RXDATA portion of the SPIBUF register is read.</b></p> <p>0 No data-length error has occurred.</p> <p>1 A data length error has occurred.</p>
23-16	LCSNR	0-FFh	<p><b>Last chip select number.</b> 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.</p>
15-0	RXDATA	0-FFFFh	<p><b>SPI receive data.</b> 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.</p>

## 23.11 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.
- In case 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 23-76](#) 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.

---

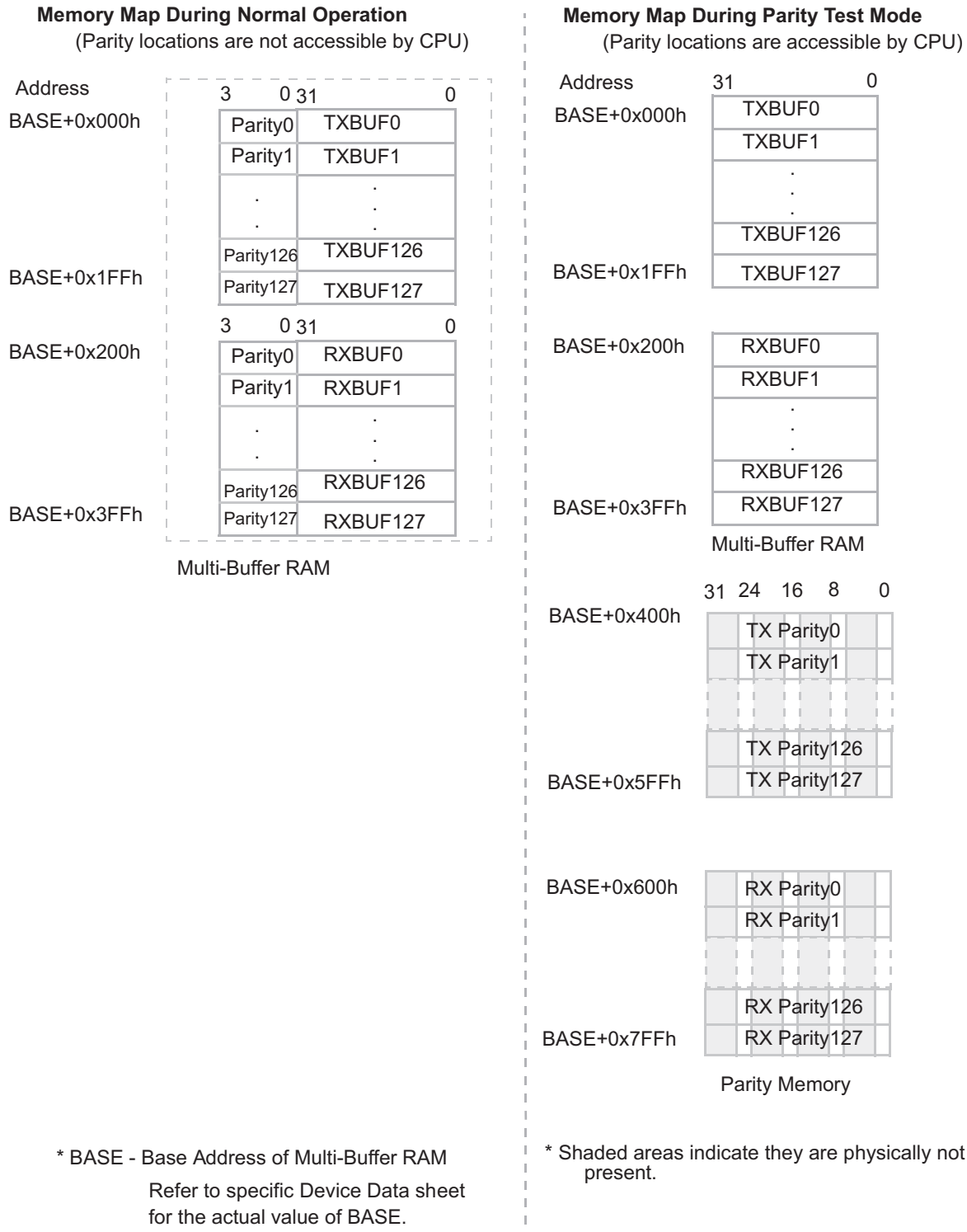
There are 4 bits of parity corresponding to each of the 32-bit multi-buffer locations. Individual bits in the parity memory are byte-addressable in parity test mode. See the example in [Figure 23-77](#) for further details.

---

**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 23-76. Memory Map for Parity Locations during Normal and Test Mode**

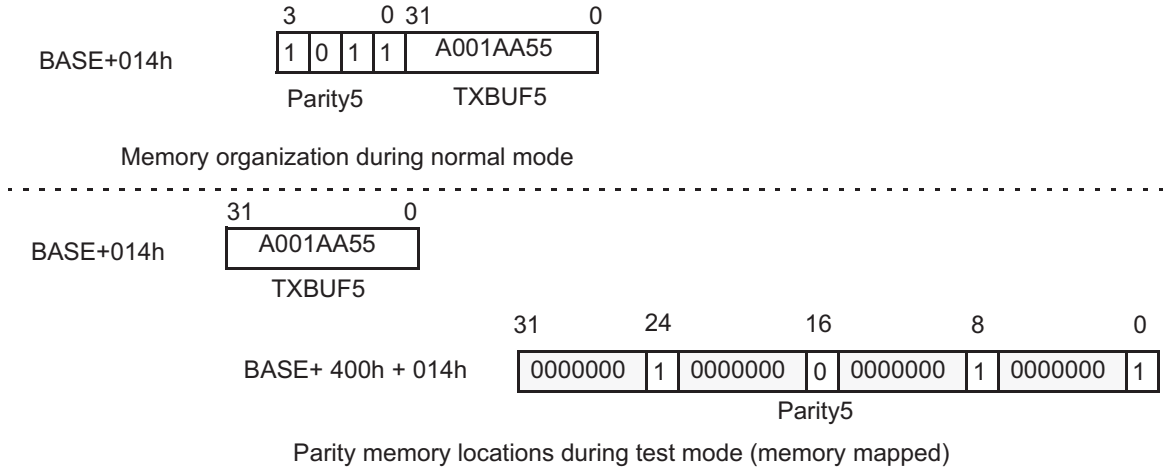


**23.11.1 Example of Parity Memory Organization**

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 23-77](#) for a diagram.

**Figure 23-77. Example of Memory Mapped Parity Locations during Test Mode**



1 Shaded areas indicate reads return 0, writes have no effect. These registers are not physically present.

**NOTE: Read Access to Parity Memory Locations**

Parity memory locations can be read even without entering into parity memory test mode. Their address remains as in memory test mode. It is only to enter parity-memory test mode to enable write access to the parity memory locations.



## 23.12 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

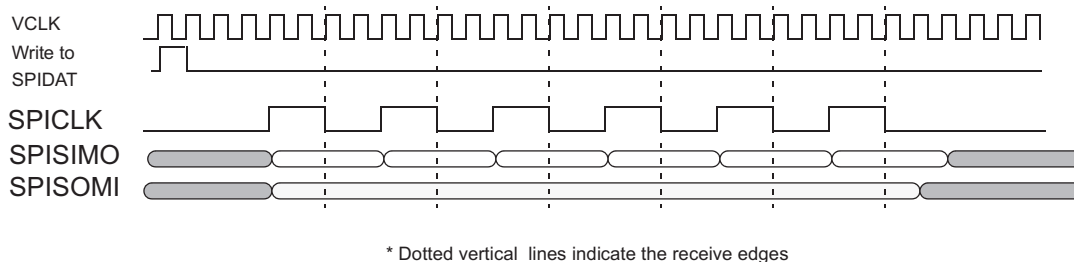
### Slave mode SPI:

- SPICLK - Input
- SPISIMO - Input
- $\overline{\text{SPISCS}}$  - Input
- SPISOMI - Output
- $\overline{\text{SPIENA}}$  - Output

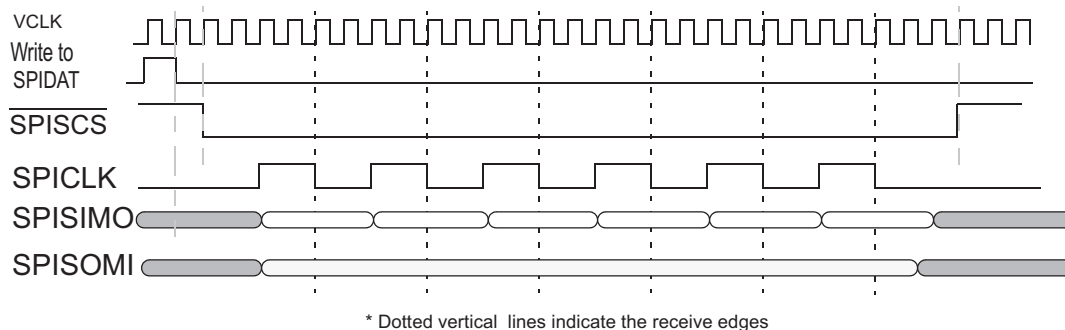
**NOTE:** All the timing diagrams given below are with Phase='0' & Polarity = '0' unless explicitly stated otherwise.

### 23.12.1 Master Mode Timings for SPI/MibSPI

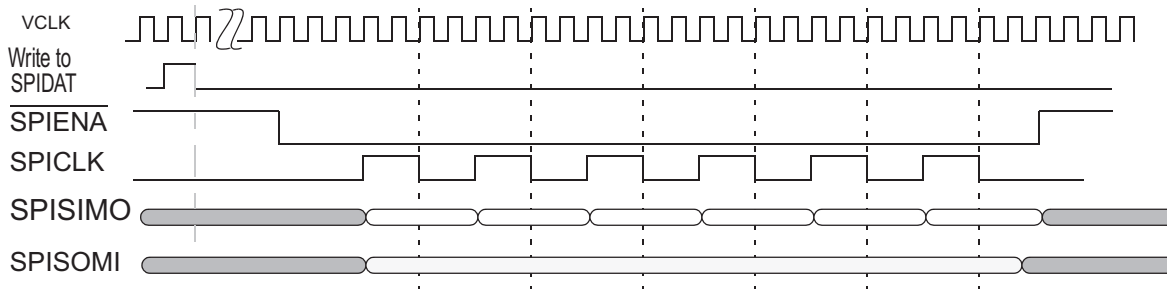
**Figure 23-78. SPI/MibSPI Pins during Master mode 3-pin configuration**



**Figure 23-79. SPI/MibSPI pins during Master mode 4-pin with  $\overline{\text{SPISCS}}$  configuration.**



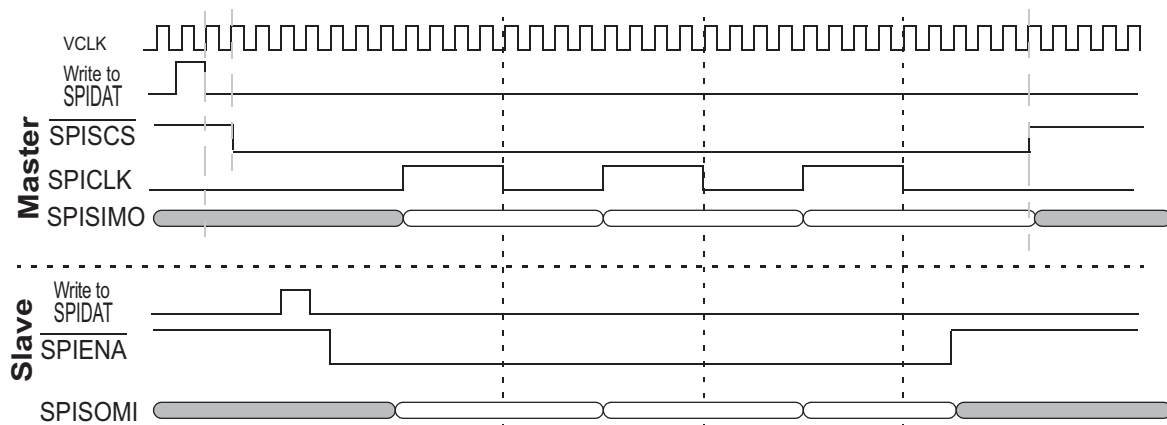
**Figure 23-80. SPI/MibSPI pins during Master mode in 4pin with  $\overline{\text{SPIEN}}\overline{\text{A}}$  configuration**



\* De-activation of  $\overline{\text{SPIEN}}\overline{\text{A}}$  pin is controlled by the Slave.

\* Dotted vertical lines indicate the receive edges

**Figure 23-81. SPI/MibSPI pins during Master/Slave mode with 5-pin configuration**

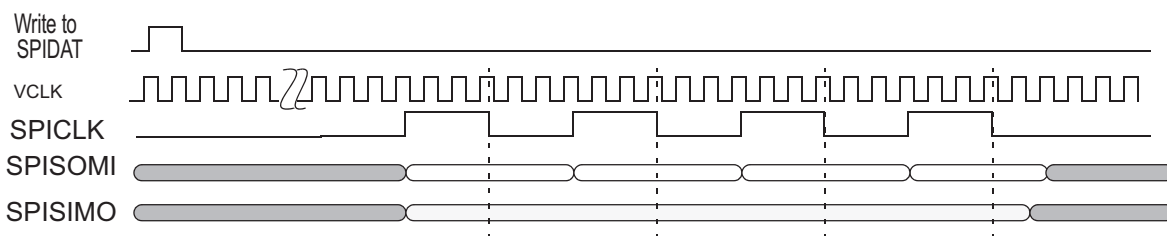


\* Dotted vertical lines indicate the receive edges for the Master

\* ENABLE\_HIGHZ is set to '0' in Slave SPI

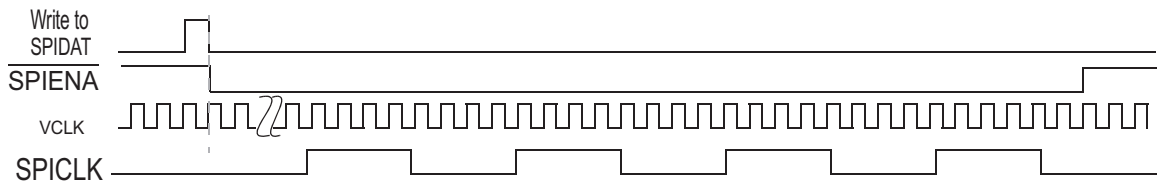
### 23.12.2 Slave Mode Timings for SPI/MibSPI

**Figure 23-82. SPI/MibSPI pins during Slave mode 3-pin configuration**



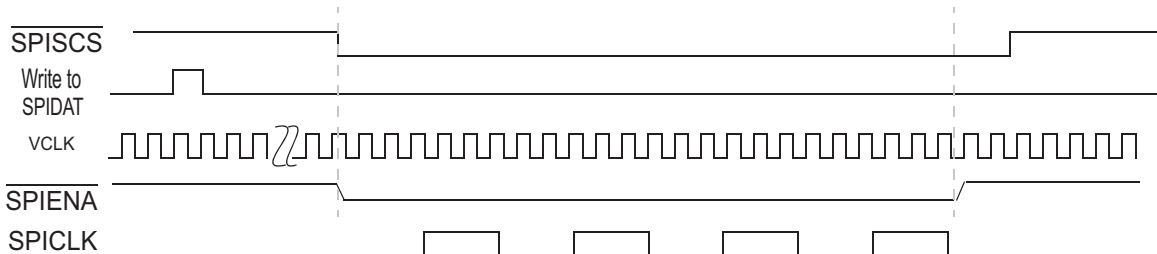
\* Dotted vertical lines indicate the receive edges

**Figure 23-83. 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

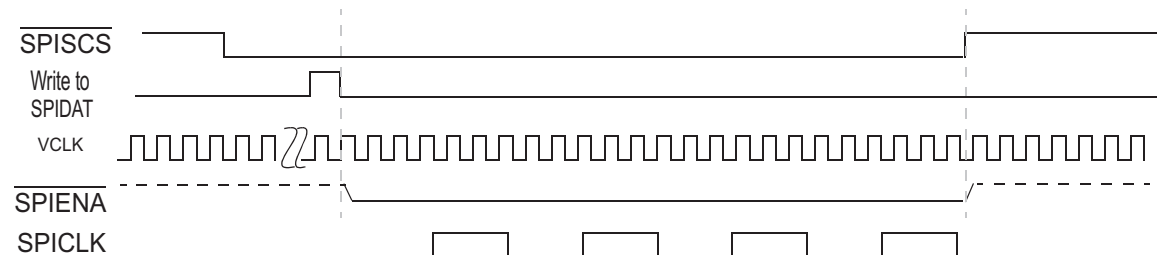
**Figure 23-84. SPI/MibSPI pins during Slave mode in 5pin configuration - (Single Slave).**



\* 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 23-85. 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

### 23.12.3 Master Mode Timing Parameter details

In case of Master, the module drives out SPICLK. It also drives out the Transmit data on SPISIMO w.r.t its internal SPICLK. In case of Master mode, the RX data on SPISOMI pin is registered w.r.t SPICLK received through Input buffer from the I/O pad.

If ChipSelect pin is functional, then Master will drive out  $\overline{\text{SPISCS}}$  pin(s) before starting the SPICLK. If  $\overline{\text{SPIENA}}$  pin is functional, then Master will wait for an active 'low' from Slave on the input pin to start the SPICLK.

**Table 23-53. Timing parameters of SPI/MibSPI pins in Master Mode**

SI No.	Delay Item	Min	Max	Note
1	Cycle Time SPICLK	2VCLK	256VCLK	SPICLK frequency is programmable in Master mode
2a	$\overline{\text{SPISCS}}$ - SPICLK edge Phase = '0' (4pin CS configuration -CSHOLD = '0')	2VCLK	$(C2TDELAY + 2) * VCLK + t_{\text{dd}}$	This timing is based on CSHOLD being '0' for the current and the previous buffer transfer

**Table 23-53. Timing parameters of SPI/MibSPI pins in Master Mode (continued)**

SI No.	Delay Item	Min	Max	Note
2b	$\overline{\text{SPISCS}}$ - SPICLK edge Phase = '0' (4pin CS configuration -CSHOLD = '1')	3VCLK	$(C2TDELAY + 3)*VCLK + t_{dd}$	This timing is based on CSHOLD being '1' for the current or the previous buffer transfer
3a	$\overline{\text{SPISCS}}$ - SPICLK edge Phase = '1' (4pin CS configuration -CSHOLD = '0')	0.5 SPICLK + 2VCLK	$(C2TDELAY + 2)*VCLK + 0.5\text{SPICLK} + t_{dd}$	This timing is based on CSHOLD being '0' for the current and the previous buffer transfer
3b	$\overline{\text{SPISCS}}$ - SPICLK edge Phase = '1' (4pin CS configuration -CSHOLD = '1')	0.5 SPICLK + 3VCLK	$(C2TDELAY + 3)*VCLK + 0.5\text{SPICLK} + t_{dd}$	This timing is based on CSHOLD being '1' for the current or the previous buffer transfer
4	Output Delay time, SPICLK TX edge to SPISIMO valid	+/- $t_{dd}$	-	Actual analog delay depends on difference of SPICLK and SPISIMO pin delays ( $t_{dd}$ ) from their generation points.
5	Output Hold time, SPICLK RX edge to SPISIMO valid	0.5SPICLK +/- $t_{dd}$	-	Actual analog delay depends on difference of SPICLK and SPISIMO pin delays ( $t_{dd}$ ) from their generation points.
6	Input Setup time requirement on SPISOMI pin with respect to (w.r.t) SPICLK	+/- $t_{pdd}$	-	The incoming data on SPISOMI pin is sampled w.r.t incoming (fed back) SPICLK. So, the difference between propagation delay ( $t_{pdd}$ ) -SPICLK pin reaching as clock and SPISOMI reaching as a data to the sampling register determines the actual setup time.
7	Input Hold time requirement on SPISOMI pin w.r.t SPICLK	+/- $t_{pdd}$	-	The incoming data on SPISOMI pin is sampled w.r.t incoming (fed back) SPICLK. So, the difference between propagation delay ( $t_{pdd}$ ) -SPICLK pin reaching as clock and SPISOMI reaching as a data to the sampling register determines the actual hold time.
8	$\overline{\text{SPIENA}}$ - SPICLK edge Phase = '0' (4pin ENA or 5pin configuration)	2VCLK	3VCLK	Delay between $\overline{\text{SPIENA}}$ assertion from the Slave to the start of first edge of SPICLK by the Master
9	$\overline{\text{SPIENA}}$ - SPICLK edge Phase = '1' (4pin ENA or 5pin configuration)	0.5SPICLK + 2VCLK	0.5SPICLK + 3VCLK	
10	SPICLK - $\overline{\text{SPISCS}}$ Phase = '0', Master	0.5SPICLK + 1VCLK	-	Delay between the last edge of SPICLK to de-activation of $\overline{\text{SPISCS}}$ pin by the Master
11	SPICLK - $\overline{\text{SPISCS}}$ Phase = '1', Master	1VCLK	-	
12	Maximum hold time on $\overline{\text{SPIENA}}$ after the final SPICLK edge supported by Master (Phase = '0')	-	0.5SPICLK + (WDELAY + 1)* VCLK	After the last edge of SPICLK, the maximum delay within which $\overline{\text{SPIENA}}$ pin should be deasserted so that the Master does not start the next buffer (if available) transfer. Slave SPI/MibSPI should deassert $\overline{\text{SPIENA}}$ before this time if it is not ready with the next TX data for transfer.
13	Maximum hold time on $\overline{\text{SPIENA}}$ after the final SPICLK edge supported by Master (Phase = '1')	-	(WDELAY + 1)* VCLK	After the last edge of SPICLK, the maximum delay within which ENA pin should be deasserted so that the Master does not start the next buffer (if available) transfer. Slave SPI/MibSPI should deassert $\overline{\text{SPIENA}}$ before this time if it is not ready with the next TX data for transfer.
14a	Minimum inactive time of $\overline{\text{SPISCS}}$ pin(s) between two data words (CSHOLD=0)	2VCLK + (WDELAY)* VCLK	-	This can be increased by using WDELAY counter. Applicable when CSHOLD is '0' for the current and the previous buffer transfers.
14b	Minimum inactive time of $\overline{\text{SPISCS}}$ pin(s) between two data words (CSHOLD=1)	1VCLK + (WDELAY)* VCLK	-	This can be increased by using WDELAY counter. This is applicable when CSHOLD is set to '1' for the current buffer and was '0' for the previous one.

**Table 23-53. Timing parameters of SPI/MibSPI pins in Master Mode (continued)**

SI No.	Delay Item	Min	Max	Note
15	Time at which $\overline{\text{SPIEN}}_A$ pin will be sampled after $\overline{\text{SPISCS}}$ 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.
$t_{dd}$ - delay differences between the two related pins, $t_{pdd}$ - propagation delay differences between the two related pins				

### 23.12.4 Slave Mode Timing Parameter details

In case of Slave mode, the module will drive only SPISOMI and SPIENA pins. All other pins are inputs to it. The RX data on SPISIMO pin will be registered w.r.t SPICLK pin. Slave will use  $\overline{\text{SPISCS}}$  pin to drive out  $\overline{\text{SPIEN}}_A$  pin if both are functional. If 4pin with  $\overline{\text{SPIEN}}_A$  is configured, then Slave will drive out active low signal on  $\overline{\text{SPIEN}}_A$  pin when a new data is written to the TX Shift Register. Irrespective of 4pin with  $\overline{\text{SPIEN}}_A$  or 5pin configuration, the Slave will deassert the  $\overline{\text{SPIEN}}_A$  pin after the last bit is received. If ENABLE\_HIGHZ (SPIINT0.24) bit is '0', the de-asserted value of  $\overline{\text{SPIEN}}_A$  pin will be '1'. Otherwise, it will depend upon the internal PullUp or PullDown resistor (if implemented) depending upon the Specification of the Chip.

**Table 23-54. Timing parameters of SPI/MibSPI pins in Slave Mode.**

SI No.	Delay Item	Min	Max	Note
1	Cycle Time of SPICLK	2VCLK	-	Slave supports any slower frequency of SPICLK above DC levels.
4	Output Delay time, SPICLK TX edge to SPISOMI valid	$t_{pd}$	-	Actual delay depends on the propagation delay of SPICLK pin to the Clock input of the register driving the TXDATA and the O/P buffer delay on SPISOMI pin.
5	Output Hold time, SPICLK RX edge to SPISOMI valid	0.5SPICLK $+t_{pd}$	-	Actual delay depends on the propagation delay of SPICLK pin to the Clock input of the register driving the TXDATA and the O/P buffer delay on SPISOMI pin.
6	Input Setup time requirement on SPISIMO pin with respect to (w.r.t) SPICLK	$\pm t_{pdd}$	-	The incoming data on SPISIMO pin is sampled w.r.t incoming SPICLK. So, the difference between propagation delay ( $t_{pdd}$ ) of both the pins determines the actual setup time.
7	Input Hold time requirement on SPISIMO pin w.r.t SPICLK	$\pm t_{pdd}$	-	The incoming data on SPISIMO pin is sampled w.r.t incoming SPICLK. So, the difference between propagation delay ( $t_{pdd}$ ) of both the pins determines the actual hold time.
16	SPICLK - $\overline{\text{SPIEN}}_A$ (4pin/5pin configuration)	1VCLK	2.5 VCLK	Delay between the final RX-edge of SPICLK to the deassertion of $\overline{\text{SPIEN}}_A$ pin by the Slave
17	$\overline{\text{SPISCS}}$ - $\overline{\text{SPIEN}}_A$ (5pin)	0	-	Delay between assertion of $\overline{\text{SPISCS}}$ by the Master to the assertion of $\overline{\text{SPIEN}}_A$ pin by the Slave (assuming Slave is ready with the TX data)
18	Minimum delay between deselecting the Slave using its $\overline{\text{SPISCS}}$ pin and clocking a different Slave	2VCLK	-	A SPICLK edge within 2VCLK of deassertion of $\overline{\text{SPISCS}}$ will be treated a valid edge in Slave mode.
19	$\overline{\text{SPISCS}}$ deassertion to SPISOMI tristate	-	1VCLK $+t_{pd}$	Slave will tristate the SPISOMI pin if it is deselected by $\overline{\text{SPISCS}}$ (when CS is functional)

## ***Serial Communication Interface (SCI)/Local Interconnect Network (LIN) Module***

This chapter describes the serial communication interface (SCI)/local interconnect network (LIN) module. The SCI/LIN is compliant to the LIN 2.1 protocol specified in the *LIN Specification Package*. This module can be configured to operate in either SCI (UART) or LIN mode.

Topic	Page
<b>24.1 Introduction and Features</b> .....	<b>1348</b>
<b>24.2 SCI Communication Formats</b> .....	<b>1353</b>
<b>24.3 SCI Interrupts</b> .....	<b>1361</b>
<b>24.4 SCI DMA Interface</b> .....	<b>1364</b>
<b>24.5 SCI Configurations</b> .....	<b>1365</b>
<b>24.6 SCI Low Power Mode</b> .....	<b>1367</b>
<b>24.7 LIN Communication Formats</b> .....	<b>1369</b>
<b>24.8 LIN Interrupts</b> .....	<b>1385</b>
<b>24.9 LIN DMA Interface</b> .....	<b>1386</b>
<b>24.10 LIN Configurations</b> .....	<b>1387</b>
<b>24.11 Low-Power Mode</b> .....	<b>1389</b>
<b>24.12 Emulation Mode</b> .....	<b>1391</b>
<b>24.13 SCI/LIN Control Registers</b> .....	<b>1392</b>
<b>24.14 GPIO Functionality</b> .....	<b>1447</b>

## 24.1 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 functionality of SCI/LIN Module. The initial part of the chapter explains about the SCI functionality and later part about the LIN functionality. Though the register are common for LIN and SCI, the register descriptions has notes to identify the register / bit usage in different modes.

### 24.1.1 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<sup>24</sup> 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.

---

### 24.1.2 LIN Features

The following are the features of the LIN module:

- Compatibility with LIN 1.3, 2.0 and 2.1 protocols
- 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

### **24.1.3 Block Diagram**

The SCI/LIN module contains core SCI block with added sub-blocks to support LIN protocol.

Three Major components of the SCI Module are:

- Transmitter
- Baud Clock Generator
- Receiver

**Transmitter (TX)** contains two major registers to perform the double- buffering:

- The transmitter data buffer register (SCITD) contains data loaded by the CPU to be transferred to the shift register for transmission.
- The transmitter shift register (SCITXSHF) loads data from the data buffer (SCITD) and shifts data onto the LINTX pin, one bit at a time.

#### **Baud Clock Generator**



- A programmable baud generator produces either a baud clock scaled from VBUSP CLK.

**Receiver (RX)** contains two major registers to perform the double- buffering:

- The receiver shift register (SCIRXSHF) shifts data in from the LINRX pin one bit at a time and transfers completed data into the receive data buffer.
- The receiver data buffer register (SCIRD) contains received data transferred from the receiver shift register

The SCI receiver and transmitter are double-buffered, and each has its own separate enable and interrupt bits. The receiver and transmitter may each be operated independently or simultaneously in full duplex mode.

To ensure data integrity, the SCI checks the data it receives for breaks, parity, overrun, and framing errors. The bit rate (baud) is programmable to over 16 million different rates through a 24-bit baud-select register. [Figure 24-1](#) 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 24-2](#) shows the SCI/LIN block diagram.

Figure 24-1. Detailed SCI Block Diagram

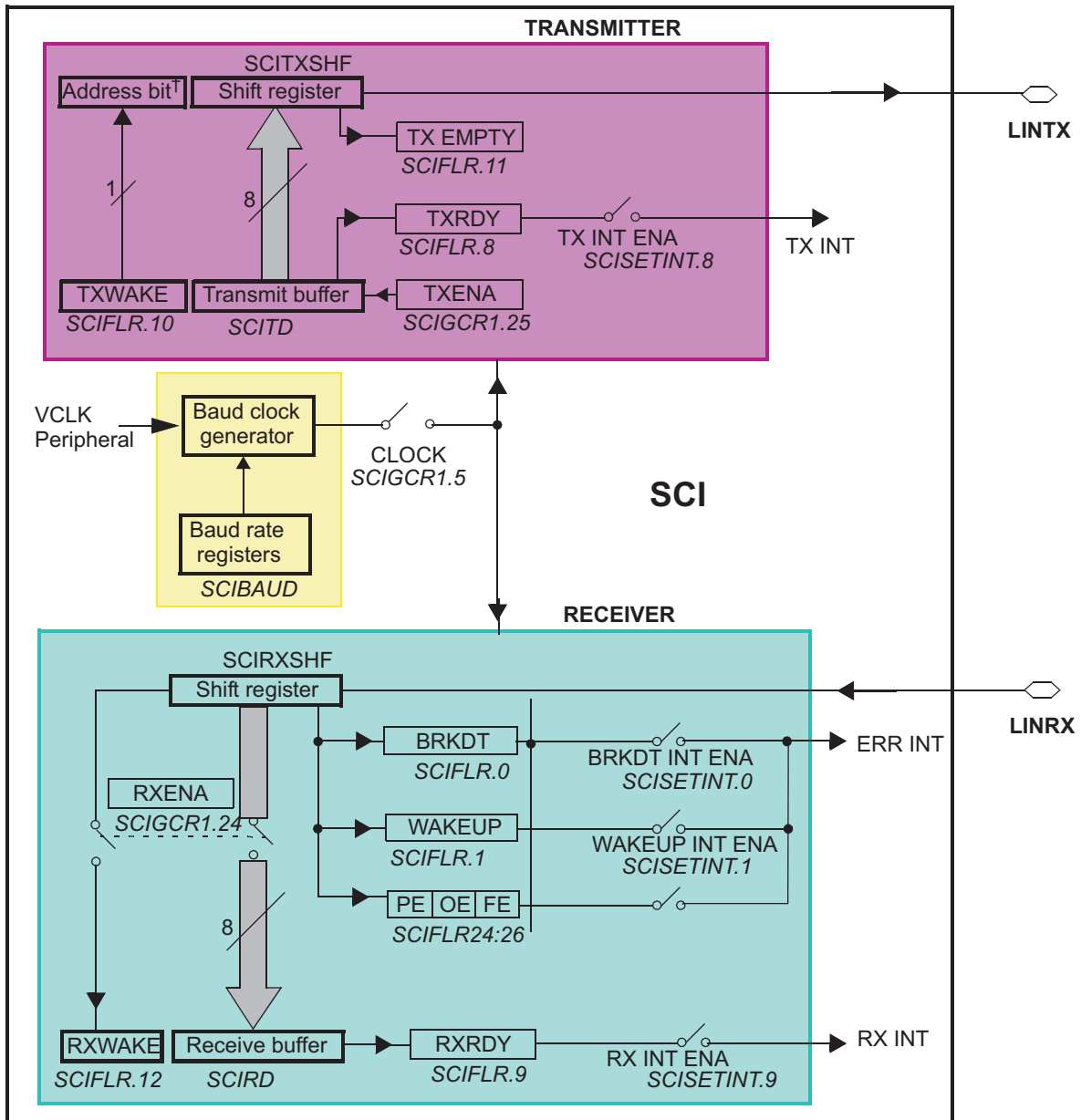
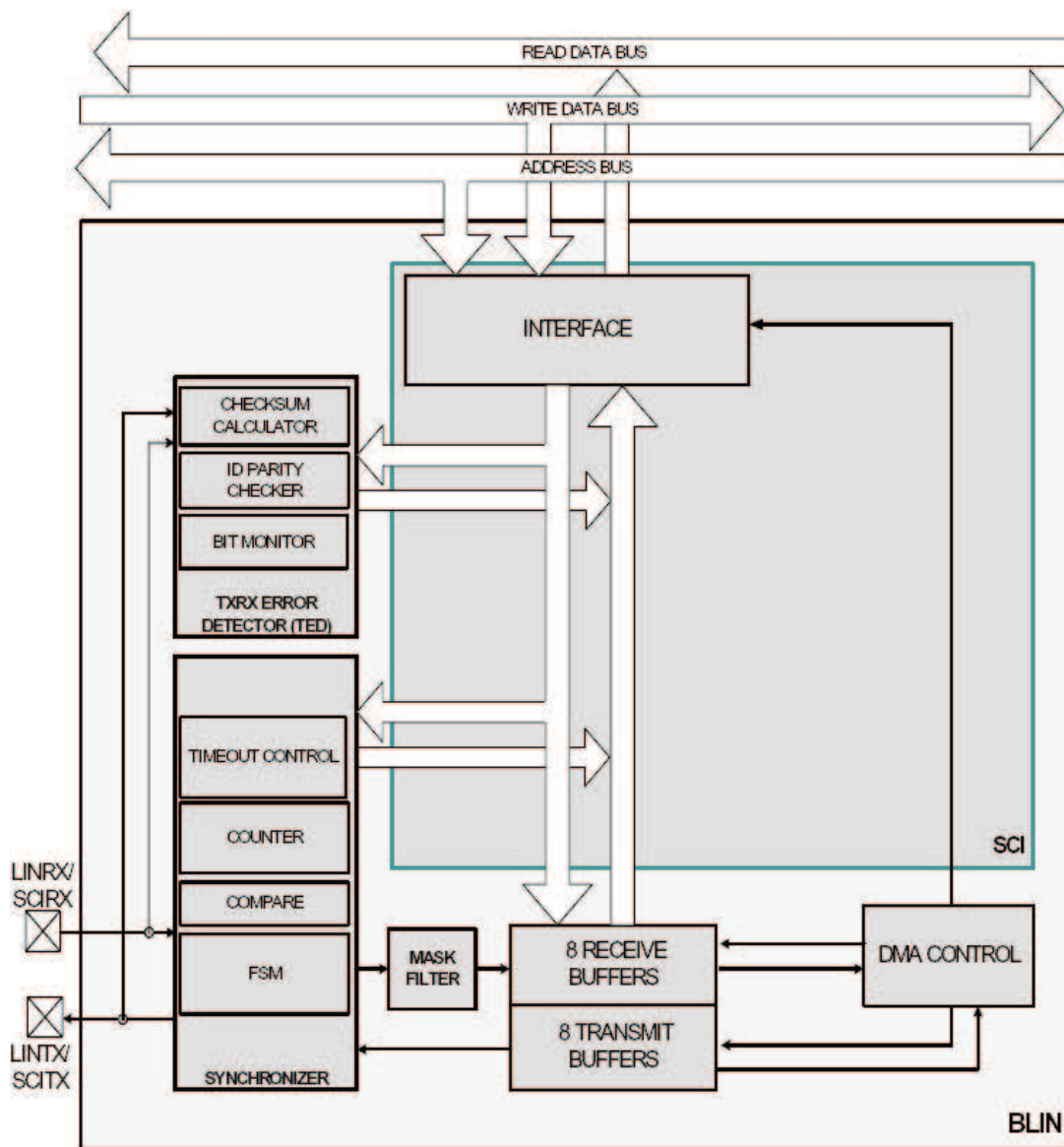


Figure 24-2. SCI/LIN Block Diagram



## 24.2 SCI Communication Formats

The SCI module can be configured to meet the requirements of many applications. Because communication formats vary depending on the specific application, many attributes of the SCI/LIN are user configurable. The list below describes these configuration options:

- SCI Frame format
- SCI Timing modes
- SCI Baud rate
- SCI Multiprocessor modes

### 24.2.1 SCI Frame Formats

The SCI uses a programmable frame format. All frames consist of the following:

- One start bit
- One to eight data bits
- Zero or one address bit
- Zero or one parity bit
- One or two stop bits

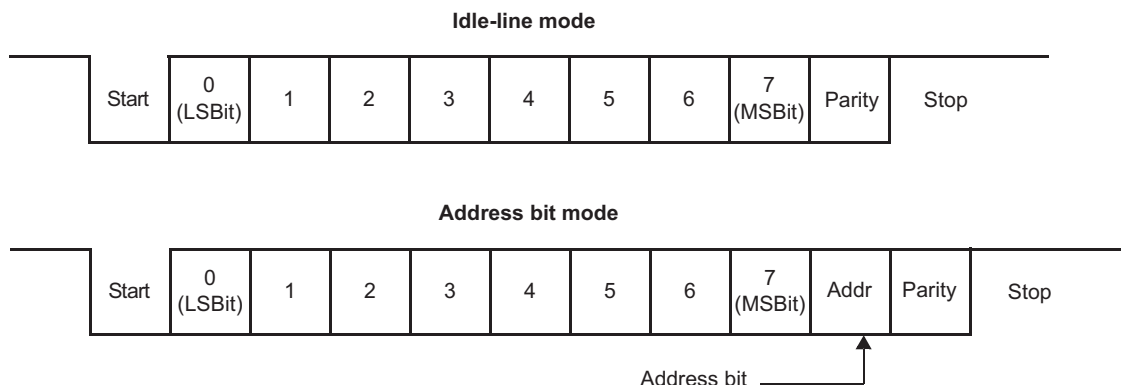
The frame format for both the transmitter and receiver is programmable through the bits in the SCIGCR1 register. Both receive and transmit data is in nonreturn to zero (NRZ) format, which means that the transmit and receive lines are at logic high when idle. Each frame transmission begins with a start bit, in which the transmitter pulls the SCI line low (logic low). Following the start bit, the frame data is sent and received least significant bit first (LSB).

An address bit is present in each frame if the SCI is configured to be in address-bit mode but is not present in any frame if the SCI is configured for idle-line mode. The format of frames with and without the address bit is illustrated in [Figure 24-3](#).

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 24-3](#) 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 24-3](#) use one stop bit per frame.

**Figure 24-3. Typical SCI Data Frame Formats**



### 24.2.2 SCI Timing Mode

The SCI can be configured to use asynchronous or isosynchronous timing using TIMING MODE bit in SCIGCR1 register.

### 24.2.2.1 Asynchronous Timing Mode

The asynchronous timing mode uses only the receive and transmit data lines to interface with devices using the standard universal asynchronous receiver-transmitter (UART) protocol.

In the asynchronous timing mode, each bit in a frame has a duration of 16 SCI baud clock periods. Each bit therefore consists of 16 samples (one for each clock period). When the SCI is using asynchronous mode, the baud rates of all communicating devices must match as closely as possible. Receive errors result from devices communicating at different baud rates.

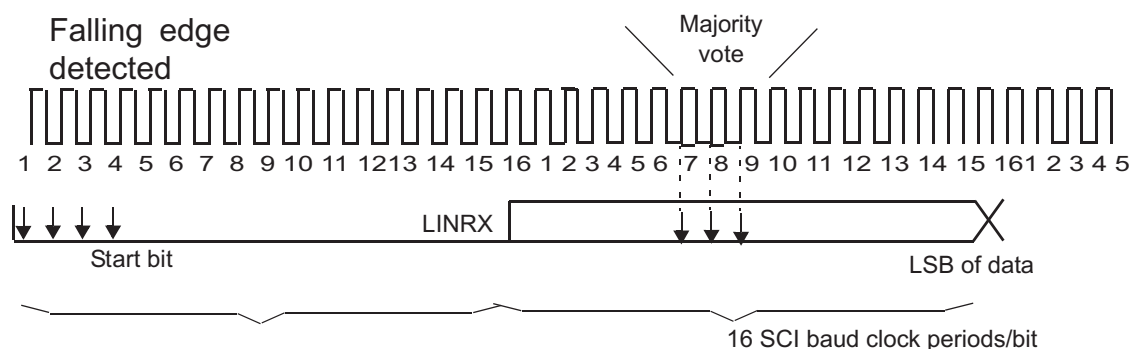
With the receiver in the asynchronous timing mode, the SCI detects a valid start bit if the first four samples after a falling edge on the LINRX pin are of logic level 0. As soon as a falling edge is detected on LINRX, the SCI assumes that a frame is being received and synchronizes itself to the bus.

To prevent interpreting noise as Start bit SCI expects LINRX line to be low for at least four contiguous SCI baud clock periods to detect a valid start bit. The bus is considered idle if this condition is not met. When a valid start bit is detected, the SCI determines the value of each bit by sampling the LINRX line value during the seventh, eighth, and ninth SCI baud clock periods. A majority vote of these three samples is used to determine the value stored in the SCI receiver shift register. By sampling in the middle of the bit, the SCI reduces errors caused by propagation delays and rise and fall times and data line noises.

Figure 24-4 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 24-4. Asynchronous Communication Bit Timing**



### 24.2.2.2 Isosynchronous Timing Mode

In isosynchronous timing mode, each bit in a frame has a duration of exactly 1 baud clock period and therefore consists of a single sample. With this timing configuration, the transmitter and receiver are required to make use of the SCICLK pin to synchronize communication with other SCI. **This mode is not supported on this device because SCICLK pin is not available.**

### 24.2.3 SCI Baud Rate

The SCI/LIN has an internally generated serial clock determined by the peripheral VCLK and the prescalers P and M in this register. The SCI uses the 24-bit integer prescaler P value of 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:

$$SCICLK \text{ Frequency} = \frac{VCLK \text{ Frequency}}{P + 1 + \frac{M}{16}}$$

$$\text{Asynchronous baud value} = \frac{SCICLK \text{ Frequency}}{16}$$

For P = 0,

$$\text{Asynchronous baud value} = \frac{VCLK \text{ Frequency}}{32} \quad (37)$$

### 24.2.3.1 Superfractional Divider, SCI Asynchronous Mode

The superfractional divider is available in SCI asynchronous mode (idle-line and address-bit mode). Building on the 4-bit fractional divider M (BRSR[27:24]), the superfractional divider uses an additional 3-bit modulating value, illustrated in Table 24-2. 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 24-1), as shown in Table 24-2.

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 24-5 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),

$$T^{i \text{ bit}} = \left[ 16 \left( P + 1 + \frac{M}{16} \right) + d \right] T_{VCLK} \quad (38)$$

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 ( $0 < d < 1$ ),

$$T^{a \text{ bit}} = \left[ 16 \left( P + 1 + \frac{M}{16} \right) + d \right] T_{VCLK} \quad (39)$$

For P = 0  $T_{bit} = 32T_{VCLK}$

**Table 24-1. Superfractional Bit Modulation for SCI Mode (Normal Configuration) <sup>(1)</sup>**

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

<sup>(1)</sup> Normal configuration = Start + 8 Data Bits + Stop Bit

**Table 24-2. Superfractional Bit Modulation for SCI Mode (Maximum Configuration) <sup>(1)</sup>**

BRS[30:28]	Start Bit	D[0]	D[1]	D[2]	D[3]	D[4]	D[5]	D[6]	D[7]	Addr	Parity	Stop0	Stop1
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

<sup>(1)</sup> Maximum configuration = Start + 8 Data Bits + Addr Bit + Parity Bit + Stop Bit 0 + Stop Bit 1

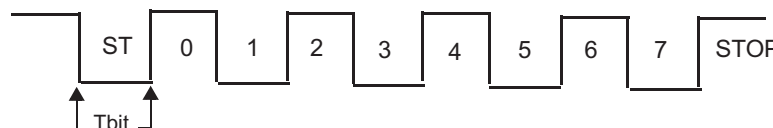
**Table 24-3. SCI Mode (Minimum Configuration) <sup>(1)</sup>**

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

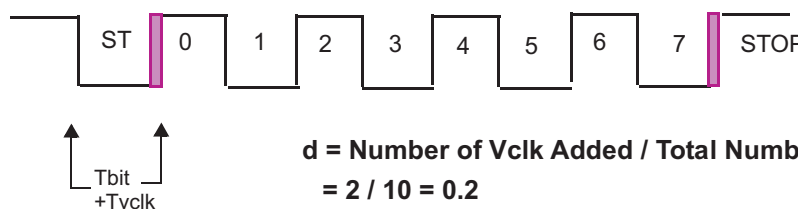
<sup>(1)</sup> Minimum configuration = Start + 1 Data Bits + Stop Bit

**Figure 24-5. Superfractional Divider Example**

**Normal Data Frame with BRS[31:28] = 0**



**Normal Data Frame with BRS[31:28] = 1**



### 24.2.4 SCI Multiprocessor Communication Modes

In some applications, the SCI may be connected to more than one serial communication device. In such a multiprocessor configuration, several frames of data may be sent to all connected devices or to an individual device. In the case of data sent to an individual device, the receiving devices must determine when they are being addressed. When a message is not intended for them, the devices can ignore the following data. When only two devices make up the SCI network, addressing is not needed, so multiprocessor communication schemes are not required.

SCI 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.

#### 24.2.4.1 Idle-Line Multiprocessor Modes

In Idle-line multiprocessor mode, a frame that is preceded by an idle period (10 or more idle bits) is an address frame. A frame that is preceded by fewer than 10 idle bits is a data frame. [Figure 24-6](#) illustrates the format of several blocks and frames with idle-line mode.

There are two ways to transmit an address frame using idle-line mode:

**Method 1:** In software, deliberately leave an idle period between the transmission of the last data frame of the previous block and the address frame of the new block.

**Method 2:** Configure the SCI to automatically send an idle period between the last data frame of the previous block and the address frame of the new block.

Although Method 1 is only accomplished by a delay loop in software, Method 2 can be implemented by using the transmit buffer and the TXWAKE bit in the following manner:

Step 1: Write a 1 to the TXWAKE bit.

Step 2: Write a dummy data value to the SCITD register. This triggers the SCI to begin the idle period as soon as the transmitter shift register is empty.

Step 3: Wait for the SCI to clear the TXWAKE flag.

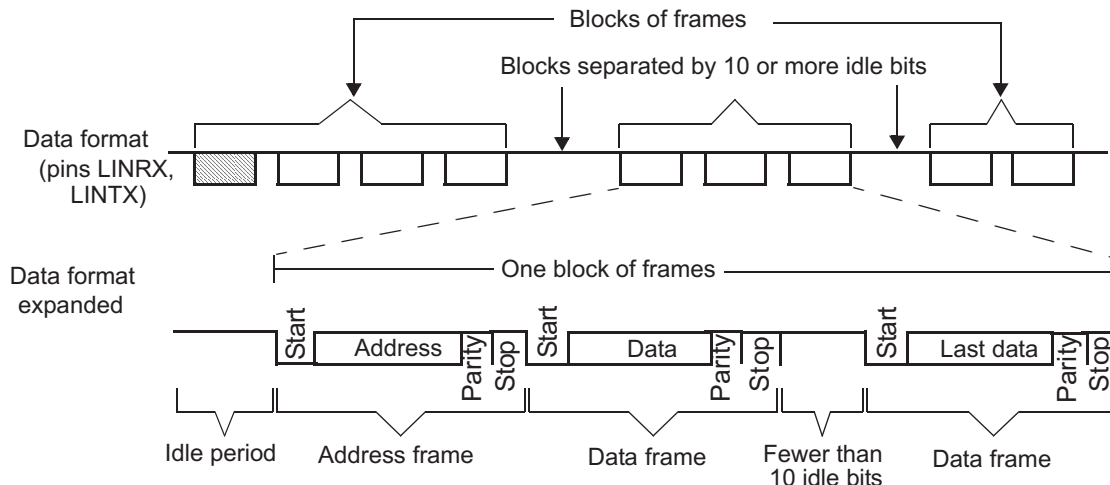
Step 4: Write the address value to SCITD.

As indicated by Step 3, software should wait for the SCI to clear the TXWAKE bit. However, the SCI clears the TXWAKE bit at the same time it sets TXRDY (that is, transfers data from SCITD into SCITXSHF). Therefore, if the TX INT ENA bit is set, the transfer of data from SCITD to SCITXSHF causes an interrupt to be generated at the same time that the SCI clears the TXWAKE bit. If this interrupt method is used, software is not required to poll the TXWAKE bit waiting for the SCI to clear it.

When idle-line multiprocessor communications are used, software must ensure that the idle time exceeds 10 bit periods before addresses (using one of the methods mentioned above), and software must also ensure that data frames are written to the transmitter quickly enough to be sent without a delay of 10 bit periods between frames. Failure to comply with these conditions will result in data interpretation errors by other devices receiving the transmission.



**Figure 24-6. Idle-Line Multiprocessor Communication Format**



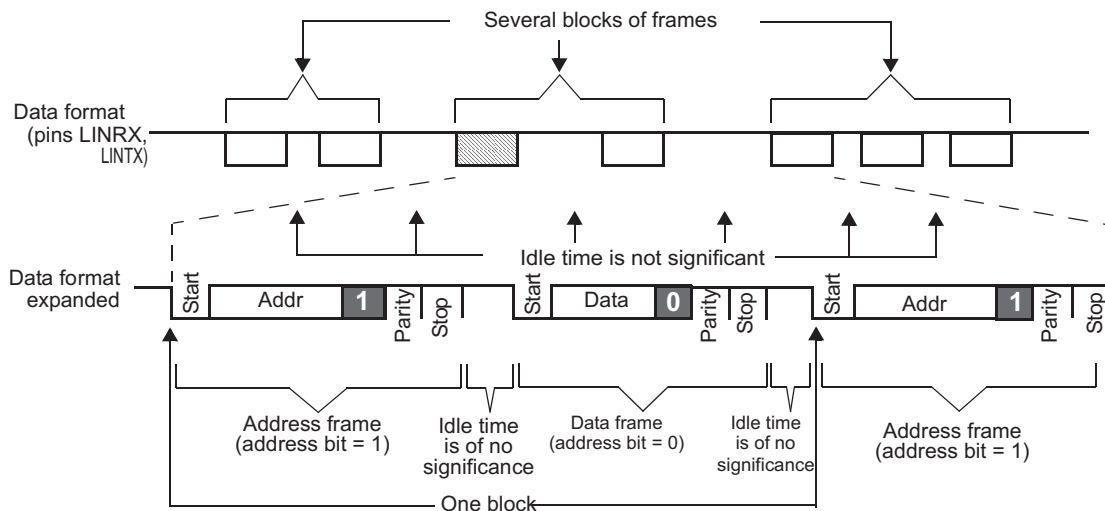
**24.2.4.2 Address-Bit Multiprocessor Mode**

In the address-bit protocol, each frame has an extra bit immediately following the data field called an address bit. A frame with the address bit set to 1 is an address frame; a frame with the address bit set to 0 is a data frame. The idle period timing is irrelevant in this mode. Figure 24-7 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 24-7. Address-Bit Multiprocessor Communication Format**



**24.2.5 SCI Multi Buffered Mode**

To reduce CPU load when Receiving or Transmitting data in interrupt mode or DMA mode, the SCI/LIN module has eight separate Receive and transmit buffers. Multi buffered mode is enabled by setting the MBUF MODE bit.

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 SCIINTVECT1 registers), 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 24-8 and Figure 24-9 shows the receive and transmit multibuffer functional block diagram.

Figure 24-8. Receive Buffers

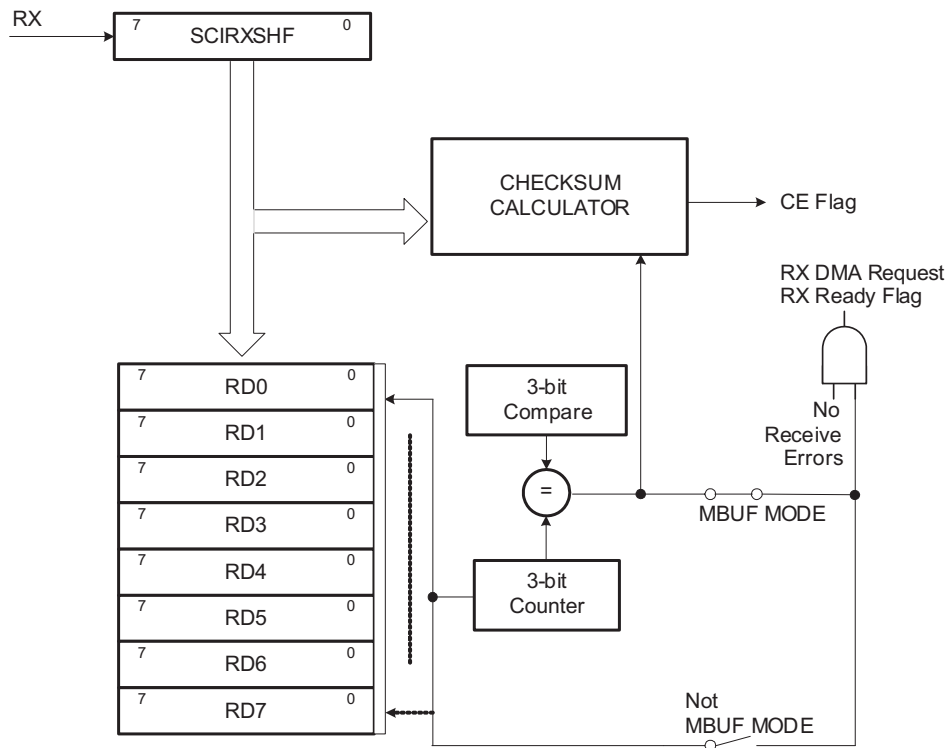
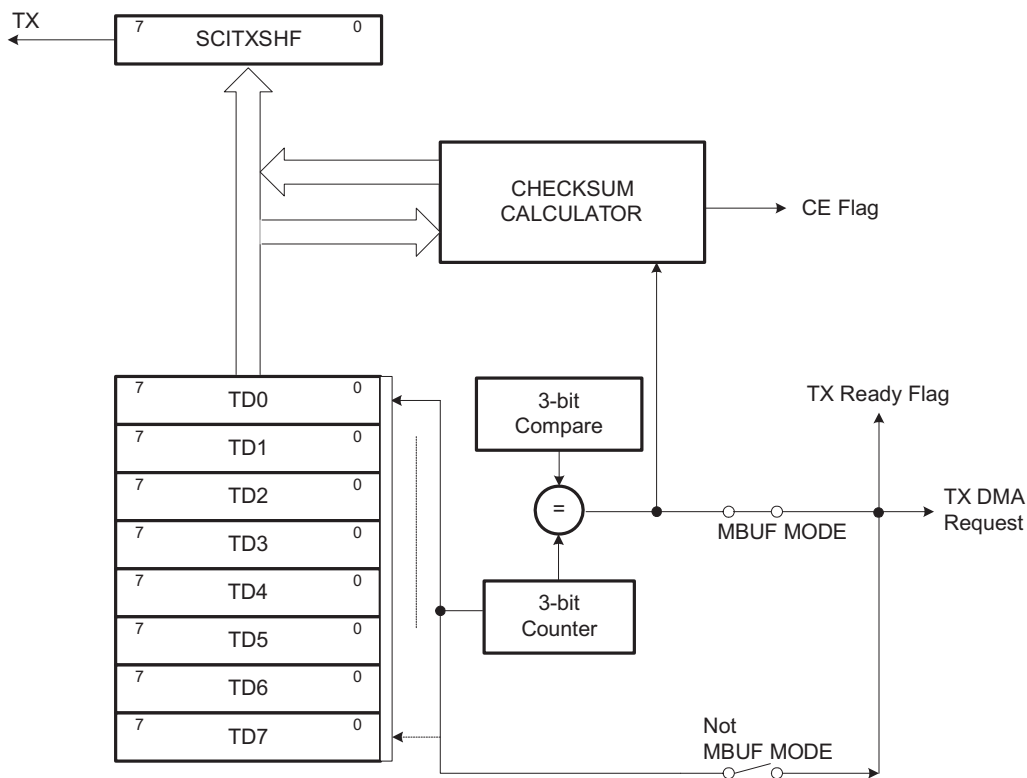


Figure 24-9. Transmit Buffers



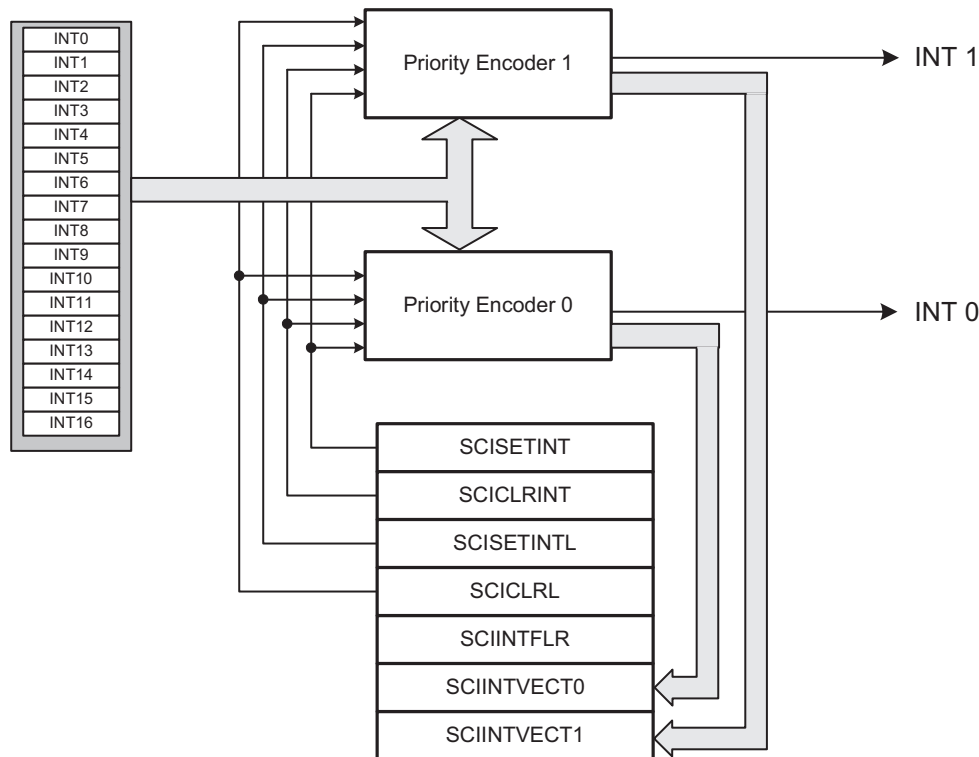
### 24.3 SCI Interrupts

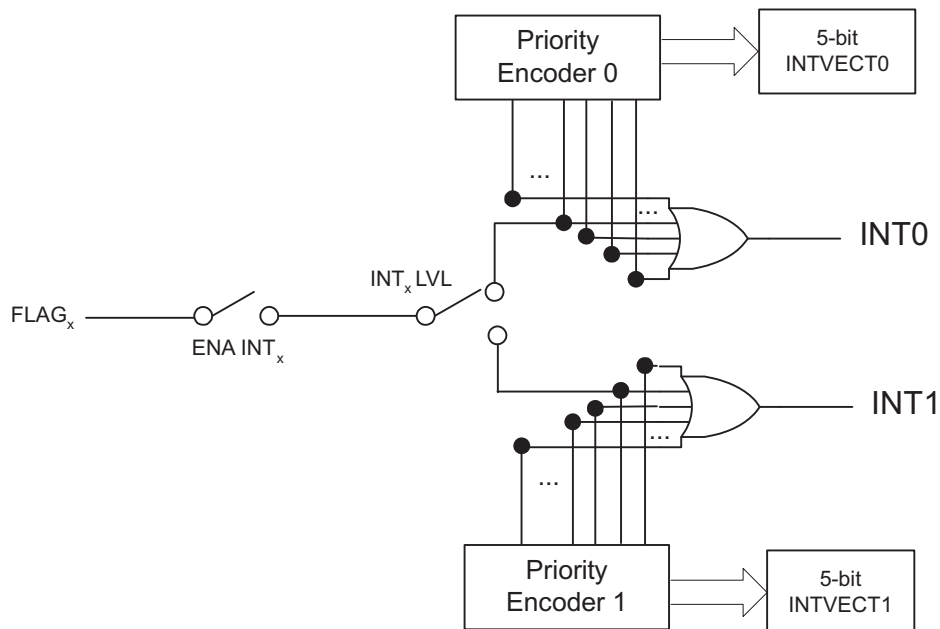
The SCI/LIN module has two interrupt lines, level 0 and level 1, to the vectored interrupt manager (VIM) module (see [Figure 24-10](#)). Two offset registers SCIINTVECT0 and SCIINTVECT1 determine which flag triggered the interrupt according to the respective priority encoders. Each interrupt condition has a bit to enable/disable the interrupt in the SCISSETINT 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. SCISSETINTLVL sets a given interrupt to level 1. 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 24-10. General Interrupt Scheme**



**Figure 24-11. Interrupt Generation for Given Flags**

### 24.3.1 Transmit Interrupt

To use transmit interrupt functionality, SET TX INT bit must be enabled and SET TX DMA bit must be cleared. The transmit ready (TXRDY) flag is set when the SCI transfers the contents of SCITD to the shift register, SCITXSHF. The TXRDY flag indicates that SCITD is ready to be loaded with more data. In addition, the SCI sets the TX EMPTY bit if both the SCITD and SCITXSHF registers are empty. If the SET TX INT bit is set, then a transmit interrupt is generated when the TXRDY flag goes high. Transmit Interrupt is not generated immediately after setting the SET TX INT bit unlike transmit DMA request. Transmit Interrupt is generated only after the first transfer from SCITD to SCITXSHF, that is first data has to be written to SCITD by the User before any interrupt gets generated. To transmit further data the user can write data to SCITD in the transmit Interrupt service routine.

Writing data to the SCITD register clears the TXRDY bit. When this data has been moved to the SCITXSHF register, the TXRDY bit is set again. The interrupt request can be suspended by setting the CLR TX INT bit; however, when the SET TX INT bit is again set to 1, the TXRDY interrupt is asserted again. The transmit interrupt request can be eliminated until the next series of values is written to SCITD, by disabling the transmitter via the TXENA bit, by a software reset SWnRST, or by a device hardware reset.

### 24.3.2 Receive Interrupt

The receive ready (RXRDY) flag is set when the SCI transfers newly received data from SCIRXSHF to SCIRD. The RXRDY flag therefore indicates that the SCI has new data to be read. Receive interrupts are enabled by the SET RX INT bit. If the SET RX INT is set when the SCI sets the RXRDY flag, then a receive interrupt is generated. The received data can be read in the Interrupt Service routine.

On a device with both SCI and a DMA controller, the bits SET RX DMA ALL and SET RX DMA must be cleared to select interrupt functionality.

### 24.3.3 WakeUp Interrupt

SCI sets the WAKEUP flag if bus activity on the RX line either prevents power-down mode from being entered, or RX line activity causes an exit from power-down mode. If enabled (SET WAKEUP INT), Wakeup interrupt is triggered once WAKEUP flag is set.

### 24.3.4 Error Interrupts

The following error detection are supported with Interrupt by the SCI module:

- Parity errors (PE)
- Frame errors (FE)
- Break Detect errors(BRKDT)
- Overrun errors (OE)
- Bit errors (BE)

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 24-4](#).

**Table 24-4. SCI/LIN Interrupts**

Offset <sup>(1)</sup>	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

<sup>(1)</sup> Offset 1 is the highest priority. Offset 16 is the lowest priority.

## 24.4 SCI DMA Interface

DMA requests for receive (RXDMA request) and transmit (TXDMA request) are available for the SCI/LIN module. The DMA transfers depending on whether multibuffer mode bit ( MBUF MODE) is enabled or not. For DMA module configuration refer

### 24.4.1 Receive DMA Requests

This DMA functionality is enabled/disabled by the CPU using the SET RX DMA/CLR RX DMA bits, respectively.

In Multi Buffered SCI mode with DMA enabled, the receiver loads the RDy buffers for each received character. RX DMA request is triggered once the last character of the programmed number of characters (LENGTH) are received and copied to the corresponding RDy buffer successfully.

If the 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.

### 24.4.2 Transmit DMA Requests

DMA functionality is enabled/disabled by the CPU with SET TX DMA/CLR TX DMA bits, respectively.

In Multi Buffered SCI mode once 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.

## 24.5 SCI Configurations

Before the SCI sends or receives data, its registers should be properly configured. Upon power-up or a system-level reset, each bit in the SCI registers is set to a default state. The registers are writable only after the RESET bit 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 24.5.1](#) and [Section 24.5.2](#))

### 24.5.1 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.

#### 24.5.1.1 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 unnecessarily 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.



### 24.5.1.2 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.

### 24.5.2 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.

#### 24.5.2.1 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.

---

#### 24.5.2.2 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.

## 24.6 SCI Low Power Mode

The SCI/LIN can be put in either local or global low-power mode. Global low-power mode is asserted by the system and is not controlled by the SCI/LIN. During global low-power mode, all clocks to the SCI/LIN are turned off so the module is completely inactive.

Local low-power mode is asserted by setting the POWERDOWN bit; setting this bit stops the clocks to the SCI/LIN internal logic and the module registers. Setting the POWERDOWN bit causes the SCI to enter local low-power mode and clearing the POWERDOWN bit causes SCI/LIN to exit from local low-power mode. All the registers are accessible during local power-down mode as any register access enables the clock to SCI for that particular access alone.

The wake-up interrupt is used to allow the SCI to exit low-power mode automatically when a low level is detected on the LINRX pin and also this clears the POWERDOWN bit. If wake-up interrupt is disabled, then the SCI/LIN immediately enters low-power mode whenever it is requested and also any activity on the LINRX pin does not cause the SCI to exit low-power mode.

---

**NOTE: Enabling Local Low-Power Mode During Receive and Transmit**

If the wake-up interrupt is enabled and low-power mode is requested while the receiver is receiving data, then the SCI immediately generates a wake-up interrupt to clear the powerdown bit and prevents the SCI from entering low-power mode and thus completes the current reception. Otherwise, if the wake-up interrupt is disabled, then the SCI completes the current reception and then enters the low-power mode.

---

### 24.6.1 Sleep Mode for Multiprocessor Communication

When the SCI receives data and transfers that data from SCIRXSHF to SCIRD, the RXRDY bit is set and if RX INT ENA is set, the SCI also generates an interrupt. The interrupt triggers the CPU to read the newly received frame before another one is received. In multiprocessor communication modes, this default behavior may be enhanced to provide selective indication of new data. When SCI receives an address frame that does not match its address, the device can ignore the data following this non-matching address until the next address frame by using sleep mode. Sleep mode can be used with both idle-line and address-bit multiprocessor modes.

If sleep mode is enabled by the SLEEP bit, then the SCI transfers data from SCIRXSHF to SCIRD only for address frames. Therefore, in sleep mode, all data frames are assembled in the SCIRXSHF register without being shifted into the SCIRD and without initiating a receive interrupt or DMA request. Upon reception of an address frame, the contents of the SCIRXSHF are moved into SCIRD, and the software must read SCIRD and determine if the SCI is being addressed by comparing the received address against the address previously set in the software and stored somewhere in memory (the SCI does not have hardware available for address comparison). If the SCI is being addressed, the software must clear the SLEEP bit so that the SCI will load SCIRD with the data of the data frames that follow the address frame.

When the SCI has been addressed and sleep mode has been disabled (in software) to allow the receipt of data, the SCI should check the RXWAKE bit (SCIFLR.12) to determine when the next address has been received. This bit is set to 1 if the current value in SCIRD is an address and set to 0 if SCIRD contains data. If the RXWAKE bit is set, then software should check the address in SCIRD against its own address. If it is still being addressed, then sleep mode should remain disabled. Otherwise, the SLEEP bit should be set again.

Following is a sequence of events typical of sleep mode operation:

- The SCI is configured and both sleep mode and receive actions are enabled.
- An address frame is received and a receive interrupt is generated.
- Software compares the received address frame against that set by software and determines that the SCI is not being addressed, so the value of the SLEEP bit is not changed.
- Several data frames are shifted into SCIRXSHF, but no data is moved to SCIRD and no receive interrupts are generated.
- A new address frame is received and a receive interrupt is generated.
- Software compares the received address frame against that set by software and determines that the

SCI is being addressed and clears the SLEEP bit.

- Data shifted into SCIRXSHF is transferred to SCIRD, and a receive interrupt is generated after each data frame is received.
- In each interrupt routine, software checks RXWAKE to determine if the current frame is an address frame.
- Another address frame is received, RXWAKE is set, software determines that the SCI is not being addressed and sets the SLEEP bit back to 1. No receive interrupts are generated for the data frames following this address frame.

By ignoring data frames that are not intended for the device, fewer interrupts are generated. These interrupts would otherwise require CPU intervention to read data that is of no significance to this specific device. Using sleep mode can help free some CPU resources.

Except for the RXRDY flag, the SCI continues to update the receiver status flags (see [Table 24-12](#)) while sleep mode is active. In this way, if an error occurs on the receive line, an application can immediately respond to the error and take the appropriate corrective action.

Because the RXRDY bit is not updated for data frames when sleep mode is enabled, the SCI can enable sleep mode and use a polling algorithm if desired. In this case, when RXRDY is set, software knows that a new address has been received. If the SCI is not being addressed, then the software should not change the value of the SLEEP bit and should continue to poll RXRDY.

## 24.7 LIN Communication Formats

The SCI/LIN module can be used in LIN mode or SCI mode. The enhancements for baud generation, DMA controls and additional receive/transmit buffers necessary for LIN mode operation are also part of the enhanced buffered SCI module. LIN mode is selected by enabling LIN MODE bit in SCIGCR1 register.

**NOTE:** The SCI/LIN is built around the SCI platform and uses a similar sampling scheme: 16 samples for each bit with majority vote on samples 8, 9, and 10.

The SCI/LIN control registers are located at the SCI/LIN base address. For a detailed description of each register, see [Section 24.13](#).

### 24.7.1 LIN Standards

For compatibility with LIN2.0 standard the following additional features are implemented over LIN1.3:

- (i) Support for LIN 2.0 checksum
- (ii) Enhanced synchronizer FSM support for frame processing
- (iii) Enhanced handling of extended frames
- (iv) Enhanced baudrate generator
- (v) Update wakeup/go to sleep

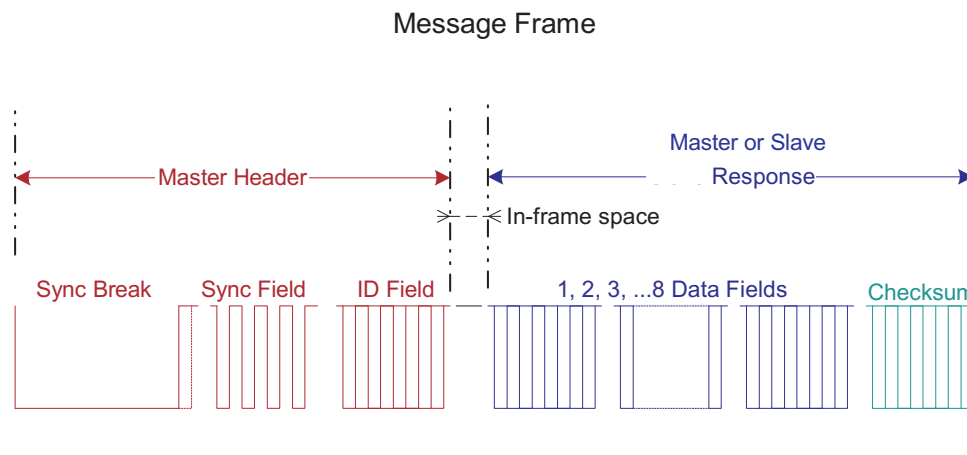
The LIN module covers the CPU performance-consuming features, defined in the *LIN Specification Package Revision 1.3 and 2.0* by hardware.

The Master Mode of LIN module is compatible with LIN 2.1 standard.

### 24.7.2 Message Frame

The LIN protocol defines a message frame format, illustrated in [Figure 24-12](#). 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 24-12. 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.

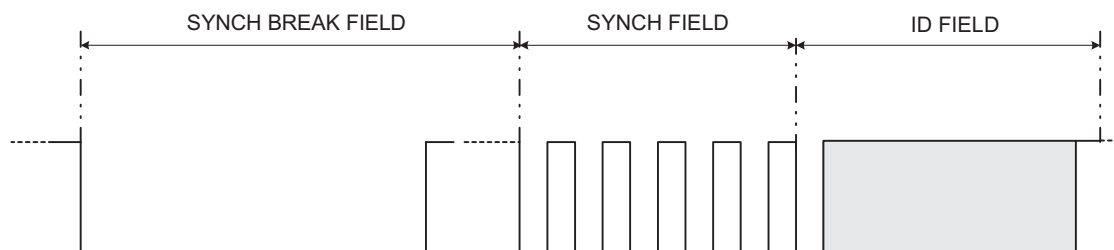
The LIN bus is a single channel wired-AND. The bus has a binary level: either dominant for a value of 0, or recessive for a value of 1.

#### 24.7.2.1 Message Header

The header of a message is initiated by a master (see [Figure 24-13](#)) 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

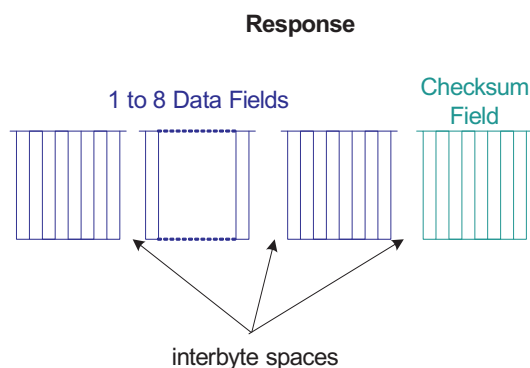
**Figure 24-13. Header 3 Fields: Synch Break, Synch, and ID**



### 24.7.2.2 Response

The format of the response is as illustrated in [Figure 24-14](#). 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 24-14. Response Format of LIN Message Frame**



The format of the response is a stream of N data fields and one checksum field. Typically N is from 1 to 8, with the exception of the extended command frames ([Section 24.7.6](#)). The length N of the response is indicated either with the optional length control bits of the ID Field (this is used in standards earlier than LIN 1.x); see [Table 24-5](#), or by LENGTH value in SCIFORMAT[18:16] register; see [Table 24-6](#). The SCI/LIN module supports response lengths from 1 to 8 bytes in compliance with LIN 2.0.

**Table 24-5. 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 24-6. Response Length with SCIFORMAT[18:16] Programming**

SCIFORMAT[18:16]	No. of Bytes
000	1
001	2
010	3

**Table 24-6. Response Length with SCIFORMAT[18:16] Programming (continued)**

SCIFORMAT[18:16]	No. of Bytes
011	4
100	5
101	6
110	7
111	8

### 24.7.3 Synchronizer

The synchronizer has three major functions in the messaging between master and slave nodes. It generates the master header data stream, it synchronizes to the LIN bus for responding, and it locally detects timeouts. A bit rate is programmed using the prescalers in the 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.

### 24.7.4 Baud Rate

The transmission baud rate of any node is configured by the CPU at the beginning; this defines the bit time  $T_{bit}$ . The bit time is derived from the fields P and M in the baud rate selection register (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:

$$P = 0, 1, 2, 3, \dots, 2^{24} - 1$$

$$M = 0, 1, 2, \dots, 15$$

$$U = 0, 1, 2, 3, 4, 5, 6, 7$$

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 TVCLK**” (with **a** = 0,1) is added to each  $T_{bit}$  as explained in [Section 24.7.4.2](#). If the ADAPT bit is set and the LIN slave is in adaptive baud rate mode, then all these divider values are automatically obtained during header reception when the synchronization field is measured.

The LIN protocol defines baud rate boundaries as follows:

$$1\text{kHz} \leq F_{LINCLK} \leq 20\text{kHz}$$

All transmitted bits are shifted in and out at  $T_{bit}$  periods.

#### 24.7.4.1 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:

For all P other than 0, and all M,

$$T_{bit} = 16 \left( P + 1 + \frac{M}{16} \right) T_{VCLK} \tag{40}$$

For P= 0 :  $T_{bit} = 32T_{VCLK}$

Therefore, the LINCLK frequency is given by

$$F_{\text{LINCLK}} = \frac{F_{\text{VCLK}}}{16(P+1 + \frac{M}{16})} \quad \text{For all } P \text{ other than zero}$$

$$F_{\text{LINCLK}} = \frac{F_{\text{VCLK}}}{32} \quad \text{For } P = 0 \quad (41)$$

#### 24.7.4.2 Superfractional Divider

The superfractional divider scheme applies to the following modes:

- LIN master mode (synch field + identifier field + response field + checksum field)
- LIN slave mode (response field + checksum field)

#### 24.7.4.3 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 24-7. 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  $T_{\text{bit}}$ .

**Table 24-7. Superfractional Bit Modulation for LIN Master Mode and Slave Mode <sup>(1)</sup>**

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)

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  $T_{\text{VCLK}}$  as follows:

For all  $P$  other than 0, and all  $M$  and  $d$  (0 or 1),

$$T^i \text{ bit} = \left[ 16 \left( P + 1 + \frac{M}{16} \right) + d \right] T_{\text{VCLK}} \quad (42)$$

For  $P = 0$   $T_{\text{bit}} = 32T_{\text{VCLK}}$

The averaged bit time is expressed in terms of  $T_{\text{VCLK}}$  as follows:

For all  $P$  other than 0, and all  $M$  and  $d$  ( $0 < d < 1$ ),

$$T^a \text{ bit} = \left[ 16 \left( P + 1 + \frac{M}{16} \right) + d \right] T_{\text{VCLK}} \quad (43)$$

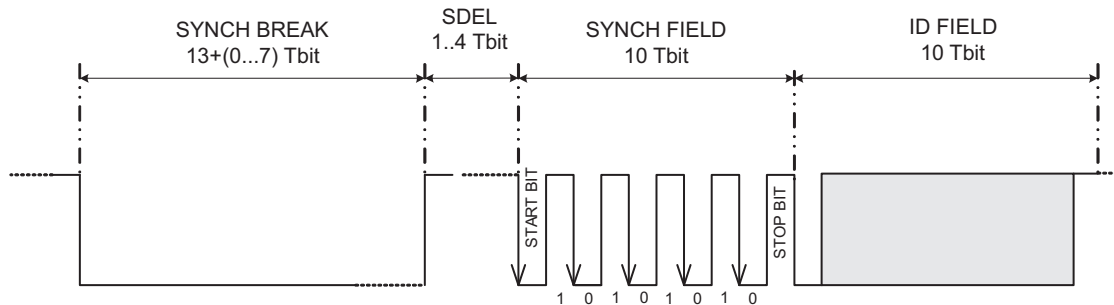
For  $P = 0$   $T_{\text{bit}} = 32T_{\text{VCLK}}$

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.

### 24.7.5 Header Generation

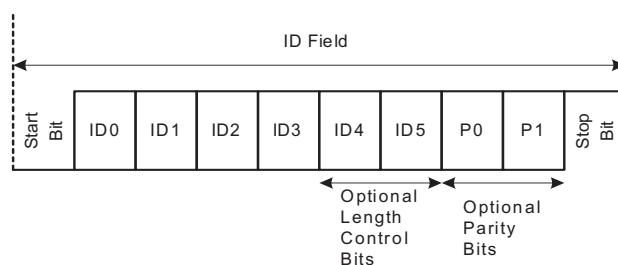
Automatic generation of the LIN protocol header data stream is supported without CPU interaction. The CPU or the DMA will trigger a message header generation and the LIN state machine will handle the generation itself. A master node initiates header generation on CPU or DMA writes to the IDBYTE in the 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 24-15.

Figure 24-15. 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 24-16 for an illustration of the ID field.

Figure 24-16. 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.

### 24.7.5.1 Event Triggered Frame Handling Proposal

The LIN 2.0 protocol uses event-triggered frames that may occasionally cause collisions. Event-triggered frames have to be handled in software.

If no slave answers to an event triggered frame header, the master node will set the NRE flag, and a NRE interrupt will occur if enabled. If a collision occurs, a frame error and checksum error may arise before the NRE error. Those errors are flagged and the appropriate interrupts will occur, if enabled.

Frame errors and checksum errors depend on the behavior and synchronization of the responding slaves. If the slaves are totally synchronized and stop transmission once the collision occurred, it is possible that only the NRE error is flagged despite the occurrence of a collision. To detect if there has been a reception of one byte before the NRE error is flagged, the BUS BUSY flag can be used as an indicator.

The bus busy flag is set on the reception of the first bit of the header and remains set until the header reception is complete, and again is set on the reception of the first bit of the response. In the case of a collision the flag is cleared in the same cycle as the NRE flag is set.

Software could implement the following sequence:

- Once the reception of the header is done (poll for RXID flag), wait for the bus busy flag to get set or NRE flag to get set.
- If bus busy flag is not set before NRE flag, then it is a true no response case (no data has been transmitted onto the bus).
- If bus busy flag gets set, then wait for NRE flag to get set or for successful reception. If NRE flag is set, then in this case a collision has occurred on the bus.

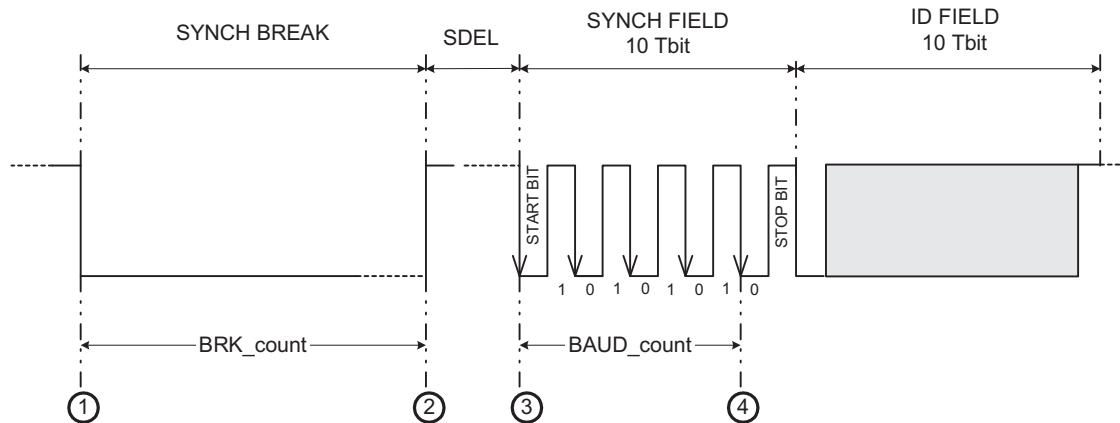
Even in the case of a collision, the received (corrupted) data is accessible in the RX buffers; registers LINRD0 and LINRD1.

### 24.7.5.2 Header Reception and Adaptive Baud Rate

A slave node baud rate can optionally be adjusted to the detected bit rate as an option to the LIN module. The adaptive baud rate option is enabled by setting the ADAPT bit. During header reception, a slave measures the baud rate during detection of the synch field. If ADAPT bit is set, then the measured baud rate is compared to the slave node's programmed baud rate and adjusted to the LIN bus baud rate if necessary.

The LIN synchronizer determines two measurements: BRK\_count and BAUD\_count ([Figure 24-17](#)). These values are always calculated during the Header reception for synch field validation ([Figure 24-18](#)).

Figure 24-17. Measurements for Synchronization



By measuring the values BRK\_count and BAUD\_count, a valid synch break sequence can be detected as described in Figure 24-18. The four numbered events in Figure 24-17 signal the start/stop of the synchronizer counter. The synchronizer counter uses VCLK as the time base.

The synchronizer counter is used to measure the synch break relative to the detecting node  $T_{bit}$ . For a slave node receiving the synch break, a threshold of 11  $T_{bit}$  is used as required by the LIN protocol. For detection of the dominant data stream of the synch break, the synchronizer counter is started on a falling edge and stopped on a rising edge of the LINRX. On detection of the synch break delimiter, the synchronizer counter value is saved and then reset.

On detection of five consecutive falling edges, the BAUD\_count is measured. Bit timing calculation and consistency to required accuracy is implemented following the recommendations of LIN revision 2.0. A slave node can calculate a single  $T_{bit}$  time by division of BAUD\_count by 8. In addition, for consistency between the detected edges the following is evaluated:

$$BAUD\_count + BAUD\_count \gg 2 + BAUD\_count \gg 3 \leq BRK\_count$$

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 24-18, 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 mistakenly be detected as a synch break.

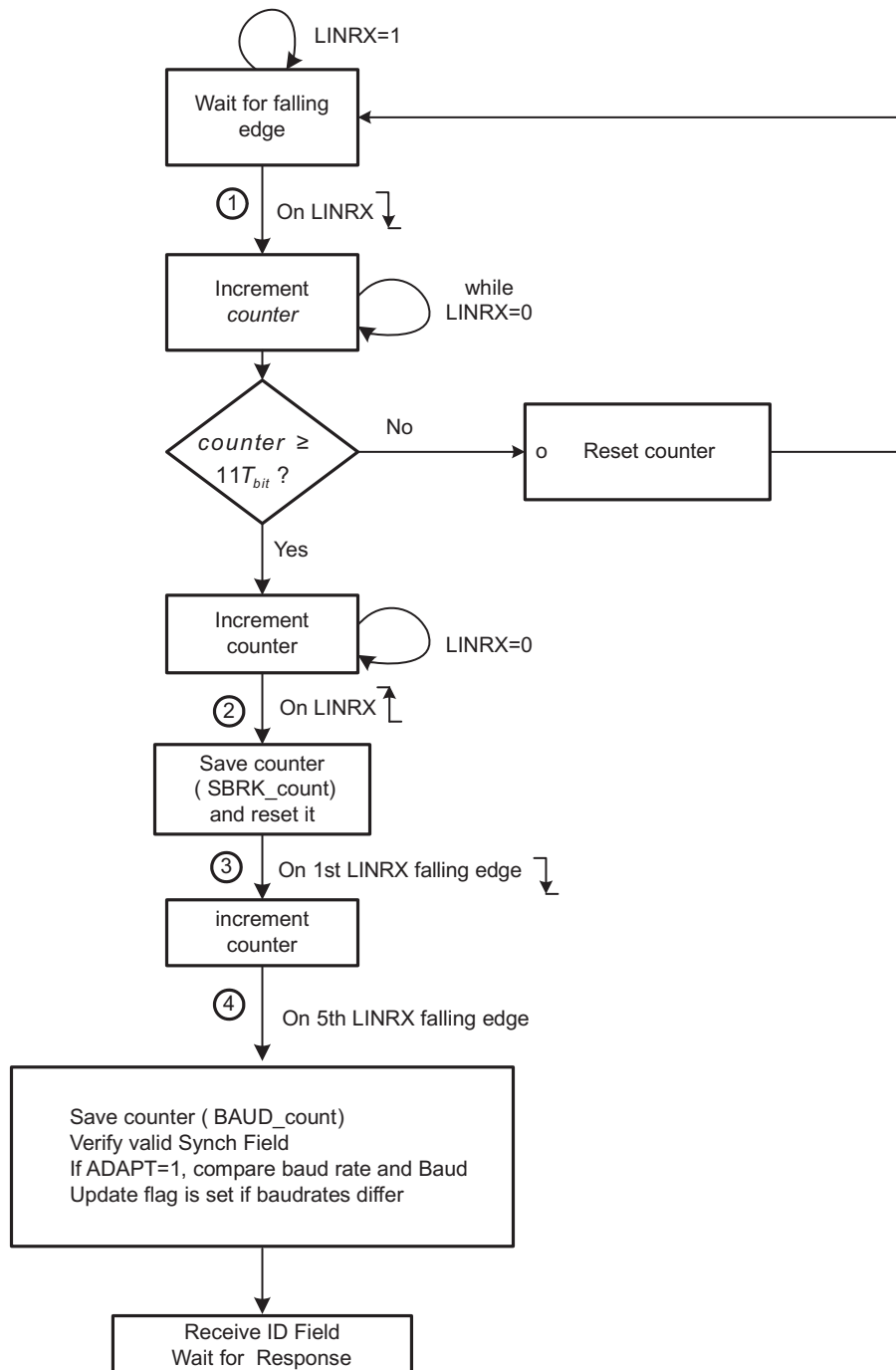
---



---

**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 24-18. 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 SCISSETINT 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.

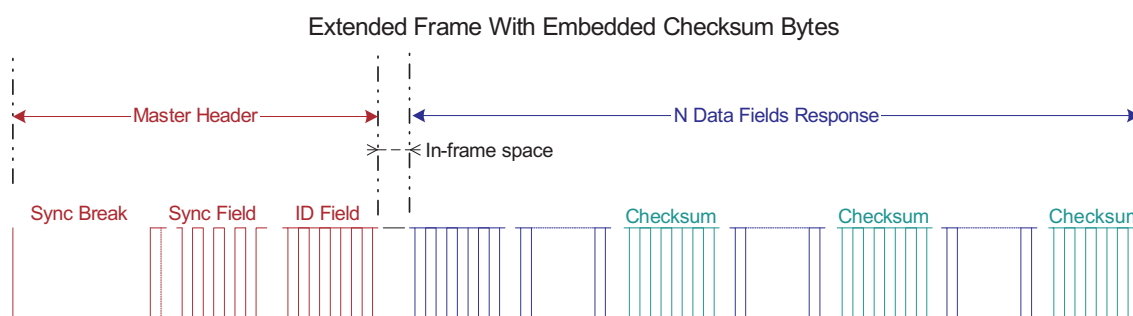
**NOTE:** When an inconsistent synch field (ISFE) error occurs, suggested action for the application is to Reset the SWnRST bit and Set the SWnRST bit to make sure that the internal state machines are back to their normal states

### 24.7.6 Extended Frames Handling

The LIN protocol 2.0 and prior includes two extended frames with identifiers 62 (user defined) and 63 (reserved extended). The response data length of the user-defined frame (ID 62, or 0x3E) is unlimited. The length for this identifier will be set at network configuration time to be shared with the LIN bus nodes.

Extended frame communication is triggered on reception of a header with identifier 0x3E; see Figure 24-19. 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 24-19. Optional Embedded Checksum in Response for Extended Frames**

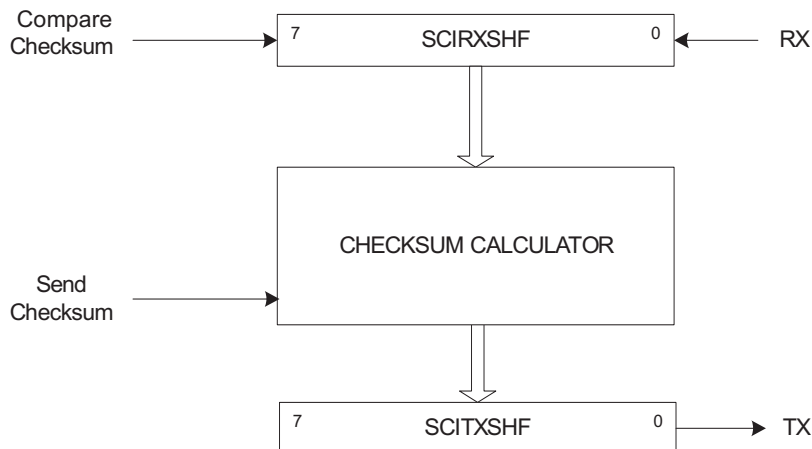


An ID interrupt will be generated (if enabled and there is a match) on reception of ID 62 (0x3E). This interrupt allows the CPU using a software counter to keep track of the bytes that are being sent out and decides when to calculate and insert a checksum byte (recommended at periodic rates). To handle 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 24-20.

**NOTE:** The LIN 2.0 enhanced checksum does not apply to the reserved identifiers. The reserved identifiers always use the classic checksum.

**Figure 24-20. Checksum Compare and Send for Extended Frames**

### 24.7.7 Timeout Control

Any LIN node listening to the bus and expecting a response initiated from a master node could flag a no-response error timeout event. The LIN protocol defines four types of timeout events, which are all handled by the hardware of the LIN module. The four LIN protocol events are:

- No-response timeout error
- Bus idle detection
- Timeout after wakeup signal
- Timeout after three wakeup signals

#### 24.7.7.1 No-Response Error (NRE)

The no-response error will occur when any node expecting a response waits for  $T_{\text{FRAME\_MAX}}$  time and the message frame is not fully completed within the maximum length allowed,  $T_{\text{FRAME\_MAX}}$ . 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:

$$T_{\text{FRAME\_MIN}} = T_{\text{HEADER\_MIN}} + T_{\text{DATA\_FIELD}} + T_{\text{CHECKSUM\_FIELD}}$$

$$= 44 + 10N$$

where  $N$  = number of data fields.

And the maximum time frame is given by:

$$T_{\text{FRAME\_MAX}} = T_{\text{FRAME\_MIN}} * 1.4$$

$$= (44 + 10N) * 1.4$$

The timeout value  $T_{\text{FRAME\_MAX}}$  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 24-8. Timeout Values in  $T_{bit}$  Units**

<b>N</b>	<b><math>T_{DATA\_FIELD}</math></b>	<b><math>T_{FRAME\_MIN}</math></b>	<b><math>T_{FRAME\_MAX}</math></b>
<b>1</b>	<b>10</b>	<b>54</b>	<b>76</b>
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

### 24.7.7.2 Bus Idle Detection

The second type of timeout can occur when a node detects an inactive LIN bus: no transitions between recessive and dominant values are detected on the bus. This happens after a minimum of 4 s (this is 80,000  $F_{LINCLK}$  cycles with the fastest bus rate of 20 kbps). If a node detects no activity in the bus as the TIMEOUT bit is set, then it can be assumed that the LIN bus is in sleep mode. Application software can use the Timeout flag to determine when the LIN bus is inactive and put the LIN into sleep mode by writing the POWERDOWN bit.

---

**NOTE:** After the timeout was flagged, a SW nRESET should be asserted before entering Low Power Mode. This is required to reset the receiver in case that an incomplete frame was on the bus before the idle period.

---

### 24.7.7.3 Timeout after Wakeup Signal and Timeout after Three Wakeup Signals

The third and fourth types of timeout are related to the wakeup signal. A node initiating a wakeup should expect a header from the master within a defined amount of time: timeout after wakeup signal. See [Section 24.11.3](#) for more details.

## 24.7.8 TXRX Error Detector (TED)

The following sources of error are detected by the TXRX error detector logic (TED). The TED logic consists of a bit monitor, an ID parity checker, and a checksum error. The following errors are detected:

- Bit errors (BE)
- Physical bus errors (PBE)
- Identifier parity errors (PE)
- Checksum errors (CE)

All of these errors (BE, PBE, PE, CE) are flagged. An interrupt for the flagged errors will be generated if enabled. A message is valid for both the transmitter and the receiver if there is no error detected until the end of the frame.

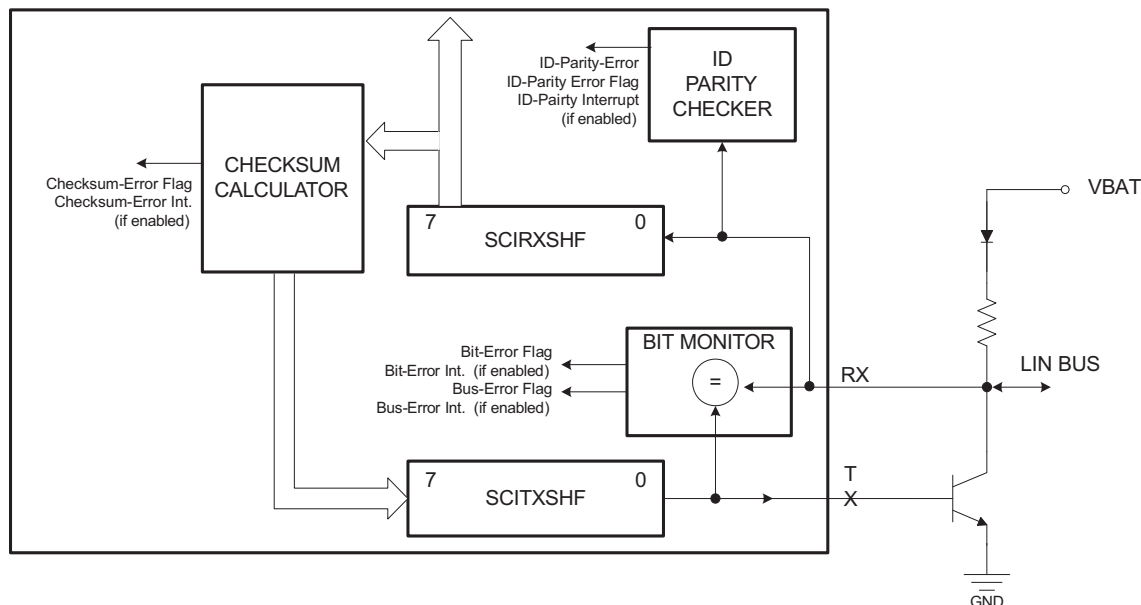
### 24.7.8.1 Bit Errors

A bit error (BE) is detected at the bit time when the bit value that is monitored is different from the bit value that is sent. A bit error is indicated by the BE flag in SCIFLR. After signaling a BE, the transmission is aborted no later than the next byte. The bit monitor ensures that the transmitted bit in LINTX is the correct value on the LIN bus by reading back on the LINRX pin as shown in [Figure 24-21](#).

---

**NOTE:** If BE Occurs due to New Header reception during a Slave Response, NRE/TIMEOUT flag will not be set for the new Frame.

---

**Figure 24-21. TXRX Error Detector**

### 24.7.8.2 Physical Bus Errors

A Physical Bus Error (PBE) has to be detected by a master if no valid message can be generated on the bus (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.

### 24.7.8.3 ID Parity Errors

If parity is enabled, an ID parity error (PE) is detected if any of the two parity bits of the sent ID byte are not equal to the calculated parity on the receiver node. The two parity bits are generated using the following mixed parity algorithm.

$$P0 = ID0 \oplus ID1 \oplus ID2 \oplus ID4 \text{ (even Parity)}$$

$$P1 = ID1 \oplus ID3 \oplus ID4 \oplus ID5 \text{ (odd Parity)} \quad (44)$$

If an ID-parity error is detected, the ID-parity error is flagged, and the received ID is not valid. See [Section 24.7.9](#) for details.

### 24.7.8.4 Checksum Errors

A checksum error (CE) is detected and flagged at the receiving end if the calculated modulo-256 sum over all received data bytes (including the ID byte if it is the enhanced checksum type) plus the checksum byte does not result in 0xFF. The modulo-256 sum is calculated over each byte by adding with carry, where the carry bit of each addition is added to the LSB of its resulting sum.

For the transmitting node, the checksum byte sent at the end of a message is the inverted sum of all the data bytes (see [Figure 24-22](#)) for classic checksum implementation. The checksum byte is the inverted sum of the identifier byte and all the data bytes (see [Figure 24-23](#)) 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 24-22. Classic Checksum Generation at Transmitting Node

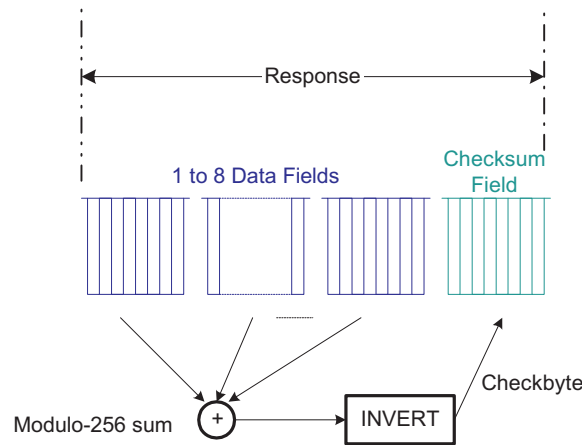
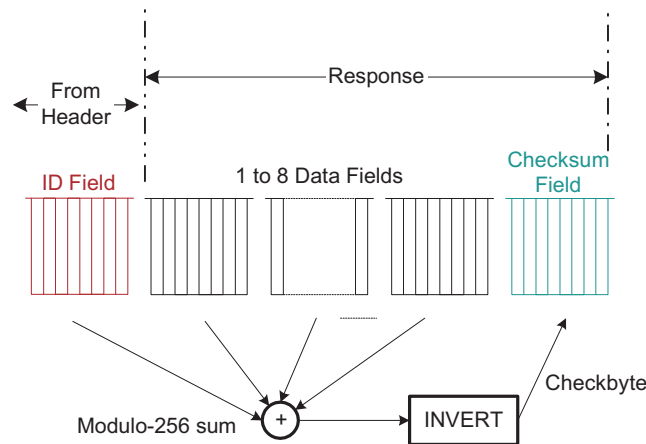


Figure 24-23. LIN 2.0-Compliant Checksum Generation at Transmitting Node



### 24.7.9 Message Filtering and Validation

Message filtering uses the entire identifier to determine which nodes will participate in a response, either receiving or transmitting a response. Therefore, two acceptance masks are used as shown in Figure 24-24. During header reception, all nodes filter the ID-Field (ID-Field is the part of the header explained in Figure 24-16) 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 24-24. 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 SCISSETINT 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:

$$(0x26 + 0x25) \oplus 0x20 = 0x07 \quad (45)$$



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.

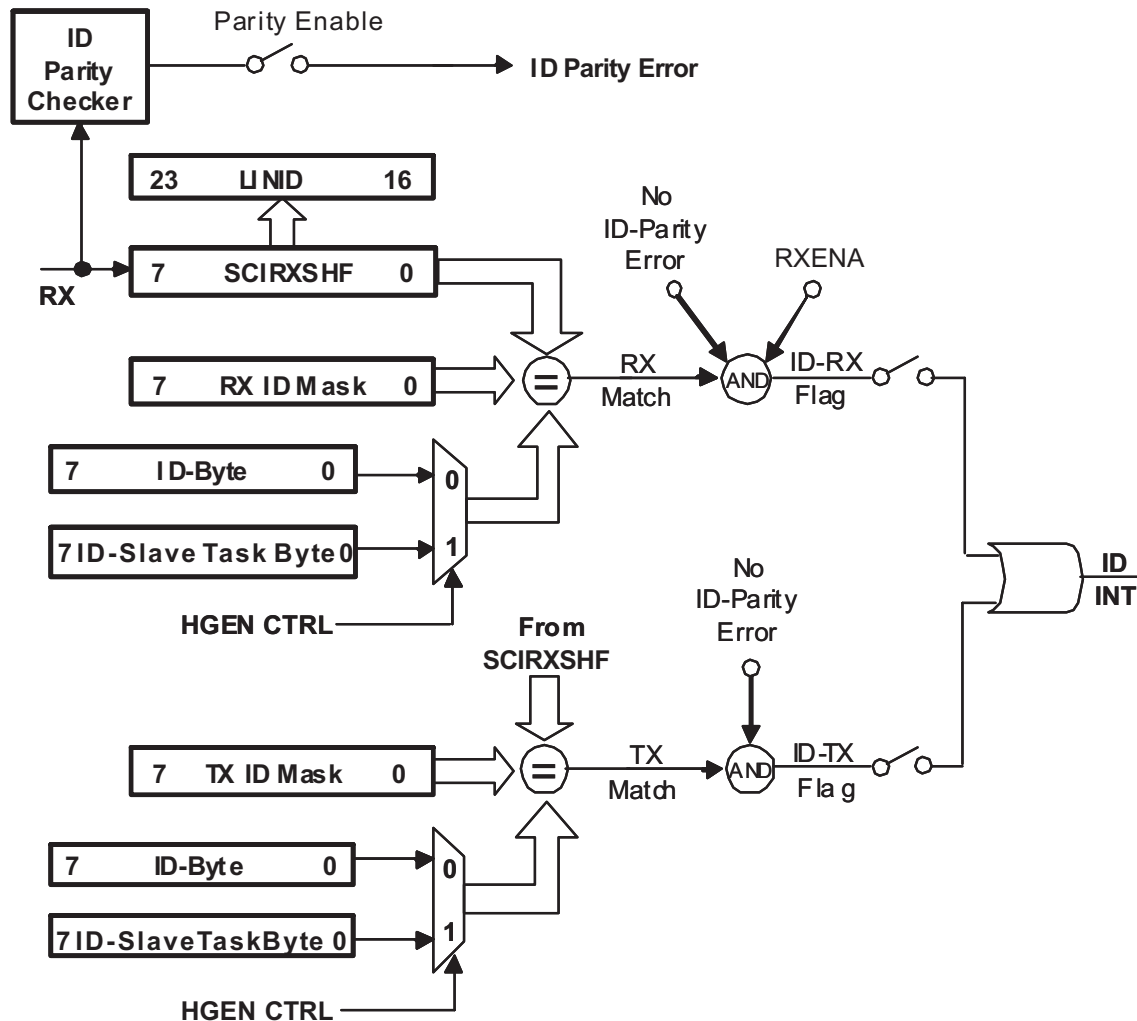
---

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 24-24. ID Reception, Filtering and Validation



### 24.7.10 Receive Buffers

To reduce CPU load when receiving a LIN N-byte (with N = 1–8) response in interrupt mode or DMA mode, the SCI/LIN module has eight receive buffers. These buffers can store an entire LIN response in the RDy receive buffers.

The checksum byte following the data bytes is validated by the internal checksum calculator. The checksum error (CE) flag indicates a checksum error and a CE interrupt will be generated if enabled in the SCISSETINT 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*, below), 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 24-8 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).

- 
- NOTE:** In multibuffer mode following are the scenarios associated with clearing the "RXRDY" flag bit:
1. The RXRDY flag cannot be cleared by reading the corresponding interrupt offset in the SCIINTVECT0/1 register.
  2. For LENGTH less than or equal to 4, Read to RD0 register will clear the "RXRDY" flag.
  3. For LENGTH greater than 4, Read to RD1 register will clear the "RXRDY" flag.
- 

### 24.7.11 Transmit Buffers

To reduce the CPU load when transmitting a LIN N-byte (with N = 1–8) response in interrupt mode or DMA mode, the SCI/LIN module has eight transmit buffers, TD0–TD7 in LINTD0 and LINTD1. With these transmit buffers, an entire LIN response field can be preloaded in the TXy transmit buffers. Optionally, a DMA transfer could be done on a byte-per-byte basis when 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 24-9 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.

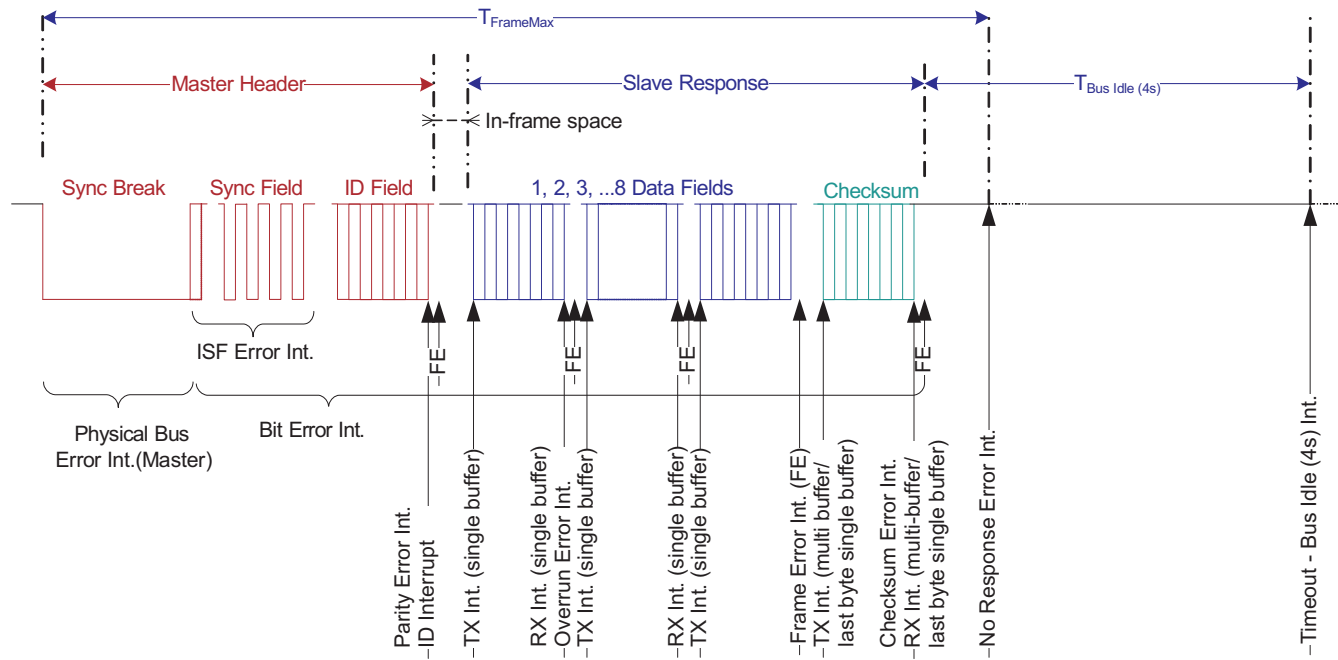
- 
- NOTE:** The transmit interrupt request can be eliminated until the next series of data is written into the transmit buffers LINTD0 and LINTD1, by disabling the corresponding interrupt via the SCICLRINT register or by disabling the transmitter via the TXENA bit.
-

## 24.8 LIN Interrupts

LIN and SCI mode have a common Interrupt block as explained in Section 24.3. There are 16 interrupt sources in the SCI/LIN module, with 8 of them being LIN mode only, as seen in Table 24-4.

A LIN message frame indicating the timing and sequence of the LIN interrupts that could occur is shown in Figure 24-25.

Figure 24-25. LIN Message Frame Showing LIN Interrupt Timing and Sequence



## 24.9 LIN DMA Interface

LIN DMA Interface uses the SCI DMA interface logic. DMA requests for receive (RXDMA request) and transmit (TXDMA request) are available for the SCI/LIN module. There are two modes for DMA transfers depending on whether multibuffer mode is enabled or not via the multibuffer enable control bit (MBUF MODE).

### 24.9.1 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.

### 24.9.2 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.

## 24.10 LIN Configurations

The following list details the configuration steps that software should perform prior to the transmission or reception of data in LIN mode. As long as 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 24.7.9](#) / [Section 24.7.10](#) / [Section 24.10.2](#))

### 24.10.1 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.

#### 24.10.1.1 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 unnecessarily 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 immediately be flagged if there is a checksum error.

### 24.10.1.2 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.

### 24.10.2 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.

#### 24.10.2.1 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.

---

#### 24.10.2.2 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.

## 24.11 Low-Power Mode

The SCI/LIN module can be put in either local or global low-power mode. Global low-power mode is asserted by the system and is not controlled by the SCI/LIN module. During global low-power mode, all clocks to the SCI/LIN are turned off so the module is completely inactive.

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.

### 24.11.1 Entering Sleep Mode

In LIN protocol, a sleep command is used to broadcast the sleep mode to all nodes. The sleep command consists of a diagnostic master request frame with identifier 0x3C (60), with the first data field as 0x00. There should be no activity in the bus once all nodes receive the sleep command: the bus is in sleep mode.

Local low-power mode is asserted by setting the POWERDOWN bit; setting this bit stops the clocks to the SCI/LIN internal logic and registers. Clearing the POWERDOWN bit causes SCI/LIN to exit from local low-power mode. All the registers are accessible during local power-down mode. If a register is accessed in low-power mode, this access results in enabling the clock to the module for that particular access alone.

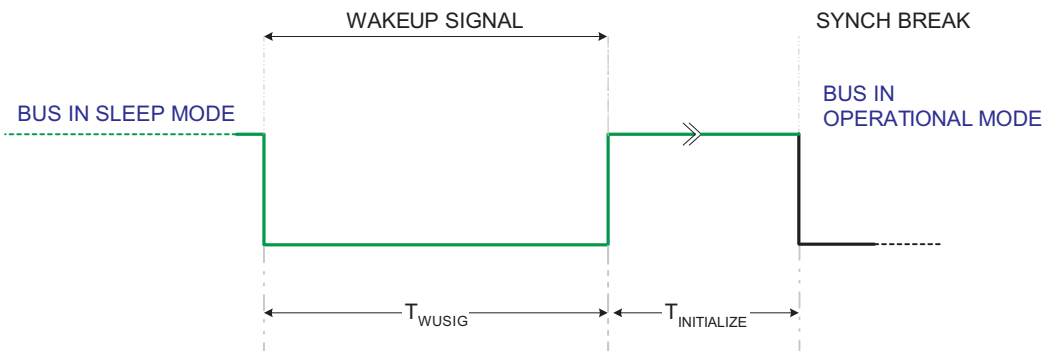
### 24.11.2 Wakeup

The wakeup interrupt is used to allow the SCI/LIN module to automatically exit low-power mode. A SCI/LIN wakeup is triggered when a low level is detected on the receive RX pin, and this clears the POWERDOWN bit.

**NOTE:** If the wakeup interrupt is disabled then the SCI/LIN enters low-power mode whenever it is requested to do so, but a low level on the receive RX pin does NOT cause the SCI/LIN to exit low-power mode.

In LIN mode, any node can terminate sleep mode by sending a wakeup signal; see [Figure 24-26](#). 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 24-26. Wakeup Signal Generation**





$$0.25\text{ms} \leq T_{\text{WUSIG}} \leq 5\text{ms} \quad (46)$$

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  $T_{\text{WUSIG}}$ . 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 SCISSETINT 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 ( $T_{\text{INITIALIZE}} < 100\text{ms}$ ) after a dominant-to-recessive edge (end-of-wakeup signal).

### 24.11.3 Wakeup Timeouts

The LIN protocol defines the following timeouts for a wakeup sequence. After a wakeup signal has been sent to the bus, all nodes wait for the master to send a header. If no synch field is detected before 150 ms (3,000 cycles at 20 kHz) after wakeup signal is transmitted, a new wakeup is sent by the same node that requested the first wakeup. This sequence is not repeated more than two times. After three attempts to wake up the LIN bus, wakeup signal generation is suspended for a 1.5 s (30,000 cycles at 20 kHz) period after three breaks.

---

**NOTE:** To achieve compatibility to LIN1.3 timeout conditions, the MBRS register must be set to assure that the LIN 2.0 (real-time-based) timings meet the LIN 1.3 bit time base. A node triggering the wakeup should set the MBRS register accordingly to meet the targeted time as  $128 \text{ Tbits} \times \text{programmed prescaler}$ .

---



---

**NOTE:** The LIN controller handles the wakeup expiration times defined by the LIN protocol with a hardware implementation.

---

## 24.12 Emulation Mode

In emulation mode, the CONT bit determines how the SCI/LIN operates when the program is suspended. The SCI/LIN counters are affected by this bit during debug mode. When set, the counters are not stopped and when cleared, the counters are stopped debug mode.

Any reads in emulation mode to a SCI/LIN register will not have any effect on the flags in the SCIFLR register.

---

**NOTE:** When emulation mode is entered during the Frame transmission or reception of the frame and CONT bit is not set, Communication is not expected to be successful. The suggested usage is to set CONT bit during emulation mode for successful communication.

---

## 24.13 SCI/LIN Control Registers

The SCI/LIN module registers are based on the SCI registers, with added functionality registers enabled by the LIN MODE bit in the SCIGCR1 register.

These registers are accessible in 8-, 16-, and 32-bit reads or writes. The SCI/LIN is controlled and accessed through the registers listed in [Table 24-9](#). Among the features that can be programmed are the LIN protocol mode, communication and timing modes, baud rate value, frame format, DMA requests, and interrupt configuration.

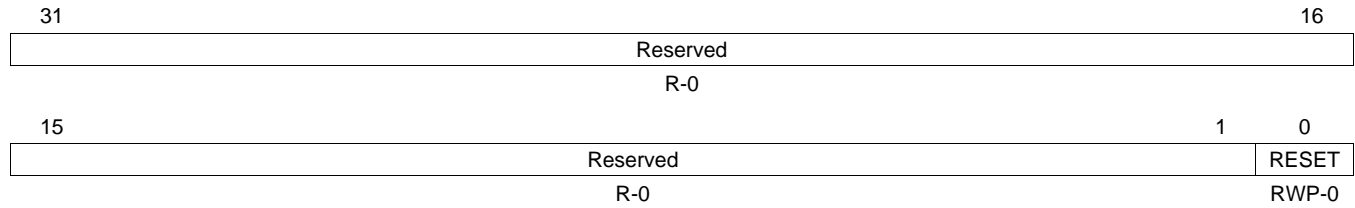
**Table 24-9. SCI/LIN Control Registers Summary**

Offset	Acronym	Register Description	Section
00	SCIGCR0	SCI Global Control Register 0	<a href="#">Section 24.13.1</a>
04h	SCIGCR1	SCI Global Control Register 1	<a href="#">Section 24.13.2</a>
08h	SCIGCR2	SCI Global Control Register 2	<a href="#">Section 24.13.3</a>
0Ch	SCISSETINT	SCI Set Interrupt Register	<a href="#">Section 24.13.4</a>
10h	SCICLEARINT	SCI Clear Interrupt Register	<a href="#">Section 24.13.5</a>
14h	SCISSETINTLVL	SCI Set Interrupt Level Register	<a href="#">Section 24.13.6</a>
18h	SCICLEARINTLVL	SCI Clear Interrupt Level Register	<a href="#">Section 24.13.7</a>
1Ch	SCIFLR	SCI Flags Register	<a href="#">Section 24.13.8</a>
20h	SCIINTVECT0	SCI Interrupt Vector Offset 0	<a href="#">Section 24.13.9</a>
24h	SCIINTVECT1	SCI Interrupt Vector Offset 1	<a href="#">Section 24.13.10</a>
28h	SCIFORMAT	SCI Format Control Register	<a href="#">Section 24.13.11</a>
2Ch	BRS	Baud Rate Selection Register	<a href="#">Section 24.13.12</a>
30h	SCIED	Receiver Emulation Data Buffer	<a href="#">Section 24.13.13.1</a>
34h	SCIRD	Receiver Data Buffer	<a href="#">Section 24.13.13.2</a>
38h	SCITD	Transmit Data Buffer	<a href="#">Section 24.13.13.3</a>
3Ch	SCIPIO0	SCI Pin I/O Control Register 0	<a href="#">Section 24.13.14</a>
40h	SCIPIO1	SCI Pin I/O Control Register 1	<a href="#">Section 24.13.15</a>
44h	SCIPIO2	SCI Pin I/O Control Register 2	<a href="#">Section 24.13.16</a>
48h	SCIPIO3	SCI Pin I/O Control Register 3	<a href="#">Section 24.13.17</a>
4Ch	SCIPIO4	SCI Pin I/O Control Register 4	<a href="#">Section 24.13.18</a>
50h	SCIPIO5	SCI Pin I/O Control Register 5	<a href="#">Section 24.13.19</a>
54h	SCIPIO6	SCI Pin I/O Control Register 6	<a href="#">Section 24.13.20</a>
58h	SCIPIO7	SCI Pin I/O Control Register 7	<a href="#">Section 24.13.21</a>
5Ch	SCIPIO8	SCI Pin I/O Control Register 8	<a href="#">Section 24.13.22</a>
60h	LINCOMPARE	LIN Compare Register	<a href="#">Section 24.13.23</a>
64h	LINRD0	LIN Receive Buffer 0 Register	<a href="#">Section 24.13.24</a>
68h	LINRD1	LIN Receive Buffer 1 Register	<a href="#">Section 24.13.25</a>
5Ch	LINMASK	LIN Mask Register	<a href="#">Section 24.13.26</a>
70h	LINID	LIN Identification Register	<a href="#">Section 24.13.27</a>
74h	LINTD0	LIN Transmit Buffer 0	<a href="#">Section 24.13.28</a>
78h	LINTD1	LIN Transmit Buffer 1	<a href="#">Section 24.13.29</a>
7Ch	MBRS	Maximum Baud Rate Selection Register	<a href="#">Section 24.13.30</a>
90h	IODFTCTRL	Input/Output Error Enable Register	<a href="#">Section 24.13.31</a>

### 24.13.1 SCI Global Control Register 0 (SCIGCR0)

The SCIGCR0 register defines the module reset. [Figure 24-27](#) and [Table 24-10](#) illustrate this register.

**Figure 24-27. SCI Global Control Register 0 (SCIGCR0) [offset = 00]**



LEGEND: R/W = Read/Write; R = Read only; RWP = Read/Write in privileged mode only; -n = value after reset

**Table 24-10. SCI Global Control Register 0 (SCIGCR0) Field Descriptions**

Bit	Field	Value	Description
31-1	Reserved	0	Read returns 0. Writes have no effect.
0	RESET	0	This bit resets the SCI/LIN module. This bit is effective in SCI and LIN mode. SCI/LIN module is in reset.
		1	SCI/LIN module is out of reset. <b>Note: Read/Write in privileged mode only.</b>

### 24.13.2 SCI Global Control Register 1 (SCIGCR1)

The SCIGCR1 register defines the frame format, protocol, and communication mode used by the SCI. Figure 24-28 and Table 24-11 illustrate this register.

**Figure 24-28. SCI Global Control Register 1 (SCIGCR1) [offset = 04h]**

31				26				25		24					
Reserved						TXENA		RXENA							
R-0						R/W-0		R/W-0							
23				18				17		16					
Reserved						CONT		LOOP BACK							
R-0						R/W-0		R/W-0							
15		14		13		12		11		10		9		8	
Reserved		STOP EXT FRAME		HGENCTRL		CTYPE		MBUFMODE		ADAPT		SLEEP			
R-0		R/WL-0		R/WL-0		R/WL-0		R/W-0		R/WL-0		R/W-0			
7		6		5		4		3		2		1		0	
SW nRST		LIN MODE		CLOCK		STOP		PARITY		PARITY ENA		TIMING MODE		COMM MODE	
R/W-0		RW-0		R/W-0		R/WC-0		R/WC-0		R/W-0		R/WC-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; WC = Write in sci-compatible mode only; -n = value after reset

**Table 24-11. SCI Global Control Register 1 (SCIGCR1) Field Descriptions**

Bit	Field	Value	Description
31-26	Reserved	0	Read returns 0. Writes have no effect.
25	TXENA	0 1	<p>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.</p> <p>0 Transfers from SCITD or TDy to SCITXSHF are disabled.</p> <p>1 Transfers from SCITD or TDy to SCITXSHF are enabled.</p> <p><b>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).</b></p>
24	RXENA	0 1	<p>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.</p> <p>0 The receiver will not transfer data from the shift buffer to the receive buffer or multi-buffers.</p> <p>1 The receiver will transfer data from the shift buffer to the receive buffer or multi-buffers.</p> <p><b>Note: Clearing RXENA stops received characters from being transferred into the receive buffer or multi-buffers, prevents the RX status flags (see Table 24-10) 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.</b></p> <p><b>Note: If RXENA is cleared before a frame is completely received, the data from the frame is not transferred into the receive buffer.</b></p> <p><b>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.</b></p>
23-18	Reserved	0	Read returns 0. Writes have no effect.

**Table 24-11. SCI Global Control Register 1 (SCIGCR1) Field Descriptions (continued)**

Bit	Field	Value	Description
17	CONT	<p>0</p> <p>1</p>	<p>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.</p> <p>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.</p> <p>When debug mode is entered, the SCI/LIN continues to operate until the current transmit and receive functions are complete.</p>
16	LOOP BACK	<p>0</p> <p>1</p>	<p>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.</p> <p>Loop back mode is disabled.</p> <p>Loop back mode is enabled.</p>
15-14	Reserved	0	Read returns 0. Writes have no effect.
13	STOP EXT FRAME	<p>0</p> <p>1</p>	<p>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.</p> <p>This bit has no effect.</p> <p>Extended frame communication will be stopped when current frame transmission/reception is completed.</p>
12	HGEN CTRL	<p>0</p> <p>1</p>	<p>HGEN control. This bit is effective in LIN mode only. This bit controls the type of mask filtering comparison</p> <p>ID filtering using the ID-BYTE field in LIN Identification Register (LINID) occurs. Mask of FFh in LIN Mask Register (LINMASK) register will result in no match.</p> <p>ID filtering uses ID-SlaveTask BYTE (recommended). Mask of FFh in LIN Mask Register (LINMASK) register will result in ALWAYS match.</p> <p><b>Note: For software compatibility with future LIN modules the HGEN CTRL bit must be set to 1, the RX ID MASK must be set to FFh and the TX ID MASK must be set to FFh.</b></p>
11	CTYPE	<p>0</p> <p>1</p>	<p>Checksum type. This bit is effective in LIN mode only. This bit controls the type of checksum to be used: classic or enhanced.</p> <p>Classic checksum is used.</p> <p>Enhanced checksum is used.</p>
10	MBUF MODE	<p>0</p> <p>1</p>	<p>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.</p> <p>The multi-buffer mode is disabled.</p> <p>The multi-buffer mode is enabled.</p>
9	ADAPT	<p>0</p> <p>1</p>	<p>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.</p> <p>Automatic baud rate adjustment is disabled.</p> <p>Automatic baud rate adjustment is enabled.</p>

**Table 24-11. SCI Global Control Register 1 (SCIGCR1) Field Descriptions (continued)**

Bit	Field	Value	Description
8	SLEEP	0 1	<p>SCI sleep. This bit is effective in SCI mode only. In a multiprocessor configuration, this bit controls the receive sleep function. Clearing this bit brings the SCI/LIN out of sleep mode.</p> <p>Sleep mode is disabled. Sleep mode is enabled.</p> <p><b>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 24-10) 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.</b></p> <p><b>Note: The SLEEP bit is <i>not</i> automatically cleared when an address byte is detected.</b></p> <p>See Section 24.6.1 for more information on using the SLEEP bit for multiprocessor communication.</p>
7	SWnRST	0 1	<p>Software reset (active low). This bit is effective in LIN and SCI modes.</p> <p>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 24-12 and Table 24-13. All affected logic is held in the reset state until a 1 is written to this bit.</p> <p>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.</p> <p><b>Note: The SCI/LIN should only be configured while SW nRESET = 0.</b></p>
6	LIN MODE	0 1	<p>LIN mode. This bit is effective in LIN and SCI mode. This bit controls the module mode of operation.</p> <p>LIN mode is disabled; SCI mode is enabled. LIN mode is enabled; SCI mode is disabled.</p>
5	CLOCK	0 1 0 1	<p>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.</p> <p><i>SCI mode</i></p> <p>The external SCICLK is the clock source.</p> <p><b>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.</b></p> <p>The internal SCICLK is the clock source.</p> <p><i>LIN mode</i></p> <p>The node is in slave mode. The node is in master mode.</p>
4	STOP	0 1	<p>SCI number of stop bits per frame. This bit is effective in SCI mode only.</p> <p>One stop bit is used. Two stop bits are used.</p> <p><b>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.</b></p>
3	PARITY	0 1	<p>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.</p> <p>Odd parity is used. Even parity is used.</p> <p><b>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.</b></p> <p><b>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.</b></p> <p><b>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.</b></p>

**Table 24-11. SCI Global Control Register 1 (SCIGCR1) Field Descriptions (continued)**

Bit	Field	Value	Description
2	PARITY ENA		Parity enable. This bit enables or disables the parity function. <i>SCI or buffered SCI mode</i>
		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. <i>LIN mode</i>
		0	ID field parity verification is disabled.
1	TIMING MODE	0	Synchronous timing is used.
		1	Asynchronous timing is used.
0	COMM MODE		SCI/LIN communication mode bit. In compatibility mode it selects the SCI communication mode. In LIN mode it selects length control option for ID-field bits ID4 and ID5. <i>SCI mode</i>
		1	Idle-line mode is used.
		0	Address-bit mode is used. <i>LIN mode</i>
		0	ID4 and ID5 are not used for length control.
		1	ID4 and ID5 are used for length control.

**Table 24-12. SCI Receiver Status Flags**

SCI Flag	Register	Bit	Value After SW nRESET <sup>(1)</sup>
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

<sup>(1)</sup> The flags are frozen with their reset value while SW nRESET = 0.

**Table 24-13. SCI Transmitter Status Flags**

SCI Flag	Register	Bit	Value After SW nRESET <sup>(1)</sup>
BE	SCIFLR	31	0
PBE	SCIFLR	30	0
TX WAKE	SCIFLR	10	0
TX EMPTY	SCIFLR	11	1
TXRDY	SCIFLR	8	1

<sup>(1)</sup> The flags are frozen with their reset value while SW nRESET = 0.



### 24.13.3 SCI Global Control Register 2 (SCIGCR2)

The SCIGCR2 register is used to send or compare a checksum byte during extended frames, to generate a wakeup and for low-power mode control of the LIN module. Figure 24-29 and Table 24-14 illustrate this register.

**Figure 24-29. SCI Global Control Register 2 (SCIGCR2) [offset = 08h]**

31	Reserved										18	17	16
										R-0		CC	SC
												R/WL- 0	R/WL- 0
15	Reserved						9	8	7			1	0
						R-0		GEN WU				POWERDO WN	
								R/W-0				R/W-0	

LEGEND: R/W = Read/Write; R = Read only; WL = Write in LIN mode only; -n = value after reset

**Table 24-14. SCI Global Control Register 2 (SCIGCR2) Field Descriptions**

Bit	Field	Value	Description
31-18	Reserved	0	Read returns 0. Writes have no effect.
17	CC	0 1	<p>Compare checksum. LIN mode only. This bit is used by the receiver for extended frames to trigger a checksum compare. The user will initiate this transaction by writing a one to this bit. CC bit has to be set only after RX_RDY flag is set for the last received data.</p> <p>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.</p> <p>During multi-buffer mode, the following scenarios are associated with the CC bit:</p> <p>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.</p> <p>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.</p> <p>c) CC bit will be auto cleared after the checkbyte has been received and compared. Checksum reception is not guaranteed if CC bit is write cleared by software during the checksum reception. See Section 24.7.6 for more details.</p> <p>0 No checksum compare will occur.</p> <p>1 Compare checksum on expected checksum byte.</p>
16	SC	0 1	<p>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 24.7.6 for more details. This byte will be cleared after the checksum byte has been transmitted.</p> <p>In non-multibuffer mode, the checksum byte will be sent after the current byte transmission.</p> <p>During multi-buffer mode, the following scenarios are associated with the SC bit:</p> <p>a) The checkbyte will be sent after the last byte count, indicated by the SCIFORMAT[18:16].</p> <p>b) Checksum will not be sent if SC is set before transmitting the very first byte (i.e., during interframe space).</p> <p>c) SC bit will be auto cleared after the checkbyte has been transmitted. Checksum transmission is not guaranteed if SC bit is write cleared by software during the checksum transmission. See Section 24.7.6 for more details.</p> <p>0 No checksum byte will be sent.</p> <p>1 A checksum byte will be sent.</p>
15-9	Reserved	0	Read returns 0. Writes have no effect.

**Table 24-14. SCI Global Control Register 2 (SCIGCR2) Field Descriptions (continued)**

Bit	Field	Value	Description
8	GEN WU		Generate wakeup signal. This bit is effective in LIN mode only. This bit controls the generation of a wakeup signal, by transmitting the TDO buffer value. The LIN protocol specifies that this signal should be a dominant for $T_{WUSIG}$ . This bit is cleared on reception of a valid synch break.
		0	No wakeup signal will be generated.
		1	The TDO buffer value will be transmitted for a wakeup signal. The bit will be cleared on a SWnRST.
7-1	Reserved	0	Read returns 0. 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 <a href="#">Section 24.11</a> 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.

### 24.13.4 SCI Set Interrupt Register (SCISSETINT)

Figure 24-30 and Table 24-15 illustrate this register. Refer Figure 24-30 for details on when different interrupt flags get set in a frame during LIN Mode.

**Figure 24-30. SCI Set Interrupt Register (SCISSETINT) [offset = 0Ch]**

31	30	29	28	27	26	25	24
SET BE INT	SET PBE INT	SET CE INT	SET ISFE INT	SET NRE INT	SET FE INT	SET OE INT	SET PE INT
R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/W-0	R/W-0	R/W-0
23				19	18	17	16
Reserved					SET RX DMA ALL	SET RX DMA	SET TX DMA
		R-0			R/WC-0	R/W-0	R/W-0
15	14	13	12		10	9	8
Reserved		SET ID INT	Reserved			SET RX INT	SET TX INT
R-0		R/WL-0		R-0		R/W-0	R/W-0
7	6	5	4	3	2	1	0
SET TOA3WUS INT	SET TOAWUS INT	Reserved	SET TIMEOUT INT	Reserved		SET WAKE UP INT	SET BRKDT INT
R/WL-0	R/WL-0	R-0	R/WL-0		R-0	R/W-0	R/WC-0

LEGEND: R/W = Read/Write; R = Read only; WL = Write in LIN mode only; WC = Write in sci-compatible mode only; -n = value after reset

**Table 24-15. SCI Set Interrupt Register (SCISSETINT) Field Descriptions**

Bit	Field	Value	Description
31	SET BE INT	0 1	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. <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect. <i>Read or write:</i> The interrupt is enabled.
30	SET PBE INT	0 1	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. <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect. <i>Read or write:</i> The interrupt is enabled.
29	SET CE INT	0 1	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. <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect. <i>Read or write:</i> The interrupt is enabled.
28	SET ISFE INT	0 1	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. <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect. <i>Read or write:</i> The interrupt is enabled.
27	SET NRE INT	0 1	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. <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect. <i>Read or write:</i> The interrupt is enabled.

**Table 24-15. SCI Set Interrupt Register (SCISSETINT) Field Descriptions (continued)**

Bit	Field	Value	Description
26	SET FE INT	0	Set framing-error interrupt. This bit is effective in LIN or SCI-compatible mode. Setting this bit enables the SCI/LIN module to generate an interrupt when a framing error occurs. <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read or write:</i> The interrupt is enabled.
25	SET OE INT	0	Set overrun-error interrupt. This bit is effective in LIN or SCI-compatible mode. Setting this bit enables the SCI/LIN module to generate an interrupt when an overrun error occurs. <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read or write:</i> The interrupt is enabled.
24	SET PE INT	0	Set parity interrupt. This bit is effective in LIN or SCI-compatible mode. Setting this bit enables the SCI/LIN module to generate an interrupt when a parity error occurs. <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read or write:</i> The interrupt is enabled.
23-19	Reserved	0	Read returns 0. Writes have no effect.
18	SET RX DMA ALL	0	Set receive DMA all. This bit is effective in SCI-compatible mode only. This bit determines if a separate interrupt is generated for the address frames sent in multiprocessor communications. When this bit is 0, RX interrupt requests are generated for address frames and DMA requests are generated for data frames. When this bit is 1, RX DMA requests are generated for both address and data frames. <i>Read:</i> The DMA request is disabled for address frames (the receive interrupt request is enabled for address frames). <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read and write:</i> The DMA request is enabled for address and data frames.
17	SET RX DMA	0	Set receiver DMA. This bit is effective in LIN or SCI-compatible mode. To enable receiver DMA requests, this bit must be set. If it is cleared, interrupt requests are generated depending on bit SCISSETINT. <i>Read:</i> The DMA request is disabled. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read and write:</i> The DMA request is enabled for address and data frames.
16	SET TX DMA	0	Set transmit DMA. This bit is effective in LIN or SCI-compatible mode. To enable DMA requests for the transmitter, this bit must be set. If it is cleared, interrupt requests are generated depending on SET TX INT bit (SCISSETINT). <i>Read:</i> Transmit DMA request is disabled. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read and write:</i> Transmit DMA request is enabled.
15-14	Reserved	0	Read returns 0. Writes have no effect.
13	SET ID INT	0	Set identification interrupt. This bit is effective in LIN mode only. This bit is set to enable an interrupt when a valid matching identifier is received. See <a href="#">Section 24.7.9</a> for more details. <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read or write:</i> The interrupt is enabled.
12-10	Reserved	0	Read returns 0. Writes have no effect.
9	SET RX INT	0	Receiver interrupt enable. This bit is effective in LIN or SCI-compatible mode. Setting this bit enables the SCI/LIN to generate a receive interrupt after a frame has been completely received and the data is being transferred from SCIRXSHF to SCIRD. <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read or write:</i> The interrupt is enabled.

**Table 24-15. SCI Set Interrupt Register (SCISSETINT) Field Descriptions (continued)**

Bit	Field	Value	Description
8	SET TX INT	0 1	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. <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect. <i>Read or write:</i> The interrupt is enabled.
7	SET TOA3WUS INT	0 1	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. <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect. <i>Read or write:</i> The interrupt is enabled.
6	SET TOAWUS INT	0 1	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. <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect. <i>Read or write:</i> The interrupt is enabled.
5	Reserved	0	Read returns 0. Writes have no effect.
4	SET TIMEOUT INT	0 1	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. <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect. <i>Read or write:</i> The interrupt is enabled.
3-2	Reserved	0	Read returns 0. Writes have no effect.
1	SET WAKEUP INT	0 1	Set wake-up interrupt. This bit is effective in LIN or SCI-compatible mode. Setting this bit enables the SCI/LIN to generate a wakeup interrupt and thereby exit low-power mode. If enabled, the wakeup interrupt is asserted when local low-power mode is requested while the receiver is busy or if a low level is detected on the LINRX pin during low-power mode. <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect. <i>Read or write:</i> The interrupt is enabled.
0	SET BRKDT INT	0 1	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. <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect. <i>Read or write:</i> The interrupt is enabled.

### 24.13.5 SCI Clear Interrupt Register (SCICLEARINT)

Figure 24-31 and Table 24-16 illustrate this register. SCICLEARINT register is used to clear the enabled interrupts without accessing SCISSETINT register.

**Figure 24-31. SCI Clear Interrupt Register (SCICLEARINT) [offset = 10h]**

31	30	29	28	27	26	25	24	
CLRBE INT	SET PBE INT	CLR CE INT	CLR ISFE INT	CLR RE INT	CLR FE INT	CLR OE INT	CLR PE INT	
R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/W-0	R/W-0	R/W-0	
23					19	18	17	16
Reserved					CLR RX DMA ALL	CLR RX DMA	CLR TX DMA	
R-0					R/WC-0	R/W-0	R/W-0	
15	14	13	12			10	9	8
Reserved		CLR ID INT	Reserved			CLR RX INT	CLR TX INT	
R-0		R/WL-0	R-0			R/W-0	R/W-0	
7	6	5	4	3	2	1	0	
CLR TOA3WUS INT	CLR TOAWUS INT	Reserved	CLR TIMEOUT INT	Reserved		CLR WAKE UP INT	CLR BRKDT INT	
R/WL-0	R/WL-0	R-0	R/WL-0	R-0		R/W-0	R/WC-0	

LEGEND: R/W = Read/Write; R = Read only; WL = Write in LIN mode only; WC = Write in sci-compatible mode only; -n = value after reset

**Table 24-16. SCI Clear Interrupt Register (SCICLEARINT) Field Descriptions**

Bit	Field	Value	Description
31	CLR BE INT		Clear bit error interrupt. This bit is effective in LIN mode only. This bit disables the bit error interrupt when set.
		0	<i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The interrupt is enabled. <i>Write:</i> 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	<i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The interrupt is enabled. <i>Write:</i> The interrupt is disabled.
29	CLR CE INT		Clear checksum-error interrupt. This bit is effective in LIN mode only. This bit disables the checksum interrupt when set.
		0	<i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The interrupt is enabled. <i>Write:</i> The interrupt is disabled.
28	CLR ISFE INT		Clear inconsistent-synch-field-error (ISFE) interrupt. This bit is effective in LIN mode only. This bit disables the ISFE interrupt when set.
		0	<i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The interrupt is enabled. <i>Write:</i> The interrupt is disabled.

**Table 24-16. SCI Clear Interrupt Register (SCICLEARINT) Field Descriptions (continued)**

Bit	Field	Value	Description
27	CLR NRE INT	0 1	Clear no-response-error interrupt. This bit is effective in LIN mode only. This bit disables the NRE interrupt when set. <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect. <i>Read:</i> The interrupt is enabled. <i>Write:</i> The interrupt is disabled.
26	CLR FE INT	0 1	Clear framing-error interrupt. This bit is effective in LIN or SCI mode. This bit disables the framing-error interrupt when set. <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect. <i>Read:</i> The interrupt is enabled. <i>Write:</i> The interrupt is disabled.
25	CLR OE INT	0 1	Clear overrun-error interrupt. This bit is effective in LIN or SCI mode. This bit disables the SCI/LIN overrun error interrupt when set. <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect. <i>Read:</i> The interrupt is enabled. <i>Write:</i> The interrupt is disabled.
24	CLR PE INT	0 1	Clear parity interrupt. This bit is effective in LIN or SCI mode. This bit disables the parity error interrupt when set. <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect. <i>Read:</i> The interrupt is enabled. <i>Write:</i> The interrupt is disabled.
23-19	Reserved	0	Read returns 0. Writes have no effect.
18	CLR RX DMA ALL	0 1	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. <i>Read:</i> 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. <i>Write:</i> Writing a 0 to this bit has no effect. <i>Read:</i> The receive DMA request for address and data frames is enabled. <i>Write:</i> The receive DMA request for address and data frames is disabled.
17	CLR RX DMA	0 1	Clear receive DMA request. This bit is effective in LIN or SCI mode. This bit disables the receive DMA request when set. <i>Read:</i> The DMA request is disabled. <i>Write:</i> Writing a 0 to this bit has no effect. <i>Read:</i> The receive DMA request is enabled. <i>Write:</i> The receive DMA request for is disabled.
16	CLR TX DMA	0 1	Clear transmit DMA request. This bit is effective in LIN or SCI mode. This bit disables the transmit DMA request when set. <i>Read:</i> Transmit DMA request is disabled. <i>Write:</i> Writing a 0 to this bit has no effect. <i>Read:</i> The transmit DMA request is enabled. <i>Write:</i> The transmit DMA request for is disabled.
15-14	Reserved	0	Read returns 0. Writes have no effect.

**Table 24-16. SCI Clear Interrupt Register (SCICLEARINT) Field Descriptions (continued)**

Bit	Field	Value	Description
13	CLR ID INT	0	Clear ID interrupt. This bit is effective in LIN mode only. This bit disables the ID interrupt when set. <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The interrupt is enabled. <i>Write:</i> The interrupt is disabled.
12-10	Reserved	0	Read returns 0. Writes have no effect.
9	CLR RX INT	0	Clear receiver interrupt. This bit is effective in LIN or SCI mode. This bit disables the receiver interrupt when set. <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The interrupt is enabled. <i>Write:</i> The interrupt is disabled.
8	CLR TX INT	0	Clear transmitter interrupt. This bit is effective in LIN or SCI mode. This bit disables the transmitter interrupt when set. <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The interrupt is enabled. <i>Write:</i> The interrupt is disabled.
7	CLR TOA3WUS INT	0	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. <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The interrupt is enabled. <i>Write:</i> The interrupt is disabled.
6	CLR TOAWUS INT	0	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. <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The interrupt is enabled. <i>Write:</i> The interrupt is disabled.
5	Reserved	0	Read returns 0. Writes have no effect.
4	CLR TIMEOUT INT	0	Clear timeout interrupt. This bit is effective in LIN mode only. This bit disables the timeout (LIN bus idle) interrupt when set. <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The interrupt is enabled. <i>Write:</i> The interrupt is disabled.
3-2	Reserved	0	Read returns 0. Writes have no effect.
1	CLR WAKEUP INT	0	Clear wake-up interrupt. This bit is effective in LIN or SCI-compatible mode. This bit disables the wakeup interrupt when set. <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The interrupt is enabled. <i>Write:</i> The interrupt is disabled.



**Table 24-16. SCI Clear Interrupt Register (SCICLEARINT) Field Descriptions (continued)**

Bit	Field	Value	Description
0	CLR BRKDT INT	0	Clear break-detect interrupt. This bit is effective in SCI-compatible mode only. This bit disables the break-detect interrupt when set. <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The interrupt is enabled. <i>Write:</i> The interrupt is disabled.

### 24.13.6 SCI Set Interrupt Level Register (SCISSETINTLVL)

Figure 24-32 and Table 24-17 illustrate this register.

**Figure 24-32. SCI Set Interrupt Level Register (SCISSETINTLVL) [offset = 14h]**

31	30	29	28	27	26	25	24
SET BE INT LVL	SET PBE INT LVL	SET CE INT LVL	SET ISFE INTLVL	SET NRE INT LVL	SET FE INT LVL	SET OE INT LVL	SET PE INT LVL
R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/W-0	R/W-0	R/W-0
23				19	18	17	16
Reserved					SET RX DMA ALL INT LVL	Reserved	
R-0				R/WC-0		R-0	
15	14	13	12	10		9	8
Reserved		SET ID INT LVL	Reserved		SET RX INT LVL	SET TX INT LVL	
R-0		R/WL-0	R-0		R/W-0	R/W-0	
7	6	5	4	3	2	1	0
SET TOA3WUS INT LVL	SET TOAWUS INT LVL	Reserved	SET TIMEOUT INT LVL	Reserved		SET WAKE UP INT LVL	SET BRKDT INT LVL
R/WL-0	R/WL-0	R-0	R/WL-0	R-0		R/W-0	R/WC-0

LEGEND: R/W = Read/Write; R = Read only; WL = Write in LIN mode only; WC = Write in sci-compatible mode only; -n = value after reset

**Table 24-17. SCI Set Interrupt Level Register (SCISSETINTLVL) Field Descriptions**

Bit	Field	Value	Description
31	SET BE INT LVL	0	Set bit error interrupt level. This bit is effective in LIN mode only. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read or write:</i> The interrupt level is mapped to the INT1 line.
30	SET PBE INT LVL	0	Set physical bus error interrupt level. This bit is effective in LIN mode only. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read or write:</i> The interrupt level is mapped to the INT1 line.
29	SET CE INT LVL	0	Set checksum-error interrupt level. This bit is effective in LIN mode only. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read or write:</i> The interrupt level is mapped to the INT1 line.
28	SET ISFE INT LVL	0	Set inconsistent-synch-field-error interrupt level. This bit is effective in LIN mode only. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read or write:</i> The interrupt level is mapped to the INT1 line.
27	SET NRE INT LVL	0	Set no-response-error interrupt level. This bit is effective in LIN mode only. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read or write:</i> The interrupt level is mapped to the INT1 line.
26	SET FE INT LVL	0	Set framing-error interrupt level. This bit is effective in LIN or SCI-compatible mode. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read or write:</i> The interrupt level is mapped to the INT1 line.

**Table 24-17. SCI Set Interrupt Level Register (SCISSETINTLVL) Field Descriptions (continued)**

Bit	Field	Value	Description
25	SET OE INT LVL	0	Set overrun-error interrupt level. This bit is effective in LIN or SCI-compatible mode. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read or write:</i> The interrupt level is mapped to the INT1 line.
24	SET PE INT LVL	0	Set parity error interrupt level. This bit is effective in LIN or SCI-compatible mode. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read or write:</i> The interrupt level is mapped to the INT1 line.
23-19	Reserved	0	Read returns 0. Writes have no effect.
18	SET RX DMA ALL LVL	0	Set receive DMA all interrupt levels. This bit is effective in SCI mode only. <i>Read:</i> The receive interrupt request for address frames is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read or write:</i> The receive interrupt request for address frames is mapped to the INT1 line.
17-14	Reserved	0	Read returns 0. Writes have no effect.
13	SET ID INT LVL	0	Set ID interrupt level. This bit is effective in LIN mode only. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read or write:</i> The interrupt level is mapped to the INT1 line.
12-10	Reserved	0	Read returns 0. Writes have no effect.
9	SET RX INT LVL	0	Set receiver interrupt level. This bit is effective in LIN or SCI-compatible mode. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read or write:</i> The interrupt level is mapped to the INT1 line.
8	SET TX INT LVL	0	Set transmitter interrupt level. This bit is effective in LIN or SCI-compatible mode. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read or write:</i> The interrupt level is mapped to the INT1 line.
7	SET TOA3WUS INT LVL	0	Set timeout after three wakeup signals interrupt level. This bit is effective in LIN mode only. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read or write:</i> The interrupt level is mapped to the INT1 line.
6	SET TOAWUS INT LVL	0	Set timeout after wakeup signal interrupt level. This bit is effective in LIN mode only. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read or write:</i> The interrupt level is mapped to the INT1 line.
5	Reserved	0	Read returns 0. Writes have no effect.
4	SET TIMEOUT INT LVL	0	Set timeout interrupt level. This bit is effective in LIN mode only. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read or write:</i> The interrupt level is mapped to the INT1 line.
3-2	Reserved	0	Read returns 0. Writes have no effect.
1	SET WAKEUP INT LVL	0	Set wake-up interrupt level. This bit is effective in LIN or SCI-compatible mode. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read or write:</i> The interrupt level is mapped to the INT1 line.

**Table 24-17. SCI Set Interrupt Level Register (SCISSETINTLVL) Field Descriptions (continued)**

Bit	Field	Value	Description
0	SET BRKDT INT LVL	0	Set break-detect interrupt level. This bit is effective in SCI-compatible mode only. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read or write:</i> The interrupt level is mapped to the INT1 line.

### 24.13.7 SCI Clear Interrupt Level Register (SCICLEARINTLVL)

Figure 24-33 and Table 24-18 illustrate this register.

**Figure 24-33. SCI Clear Interrupt Level Register (SCICLEARINTLVL) [offset = 18h]**

31	30	29	28	27	26	25	24
CLR BE INT LVL	CLR PBE INT LVL	CLR CE INT LVL	CLR ISFE INTLVL	CLR NRE INT LVL	CLR FE INT LVL	CLR OE INT LVL	CLR PE INT LVL
R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/W-0	R/W-0	R/W-0
23	Reserved			19	18	17	16
R-0					CLR RX DMA ALL INT LVL	Reserved	
R-0				R/WC-0		R-0	
15	14	13	12	10		9	8
Reserved		CLR ID IX INT LVL	Reserved			CLR ID RX INT LVL	CLR TX INT LVL
R-0		R/WL-0	R-0		R/W-0	R/W-0	
7	6	5	4	3	2	1	0
CLR TOA3WUS INT LVL	CLR TOAWUS INT LVL	Reserved	CLR TIMEOUT INT LVL	Reserved		CLR WAKE UP INT LVL	CLR BRKDT INT LVL
R/WL-0	R/WL-0	R-0	R/WL-0	R-0		R/W-0	R/WC-0

LEGEND: R/W = Read/Write; R = Read only; WL = Write in LIN mode only; WC = Write in sci-compatible mode only; -n = value after reset

**Table 24-18. SCI Clear Interrupt Level Register (SCICLEARINTLVL) Field Descriptions**

Bit	Field	Value	Description
31	CLR BE INT LVL	0	Clear bit error interrupt. This bit is effective in LIN mode only. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The interrupt level is mapped to the INT1 line. <i>Write:</i> The interrupt level is mapped to the INT0 line.
30	CLR PBE INT LVL	0	Clear physical bus error interrupt. This bit is effective in LIN mode only. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The interrupt level is mapped to the INT1 line. <i>Write:</i> The interrupt level is mapped to the INT0 line.
29	CLR CE INT LVL	0	Clear checksum-error interrupt. This bit is effective in LIN mode only. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The interrupt level is mapped to the INT1 line. <i>Write:</i> The interrupt level is mapped to the INT0 line.
28	CLR ISFE INT LVL	0	Clear inconsistent-synch-field-error (ISFE) interrupt. This bit is effective in LIN mode only. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The interrupt level is mapped to the INT1 line. <i>Write:</i> The interrupt level is mapped to the INT0 line.
27	CLR NRE INT LVL	0	Clear no-response-error interrupt. This bit is effective in LIN mode only. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The interrupt level is mapped to the INT1 line. <i>Write:</i> The interrupt level is mapped to the INT0 line.

**Table 24-18. SCI Clear Interrupt Level Register (SCICLEARINTLVL) Field Descriptions (continued)**

Bit	Field	Value	Description
26	CLR FE INT LVL	0	Clear framing-error interrupt. This bit is effective in LIN or SCI-compatible mode. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The interrupt level is mapped to the INT1 line. <i>Write:</i> The interrupt level is mapped to the INT0 line.
25	CLR OE INT LVL	0	Clear overrun-error interrupt. This bit is effective in LIN or SCI-compatible mode. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The interrupt level is mapped to the INT1 line. <i>Write:</i> The interrupt level is mapped to the INT0 line.
24	CLR PE INT LVL	0	Clear parity interrupt. This bit is effective in LIN or SCI-compatible mode. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The interrupt level is mapped to the INT1 line. <i>Write:</i> The interrupt level is mapped to the INT0 line.
23-19	Reserved	0	Read returns 0. Writes have no effect.
18	CLR RX DMA ALL LVL	0	Clear receive DMA interrupt level. This bit is effective in SCI-compatible mode only. <i>Read:</i> The receive interrupt request for address frames is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The receive interrupt request for address frames is mapped to the INT1 line. <i>Write:</i> The receive interrupt request for address frames is mapped to the INT0 line.
17-14	Reserved	0	Read returns 0. Writes have no effect.
13	CLR ID INT LVL	0	Clear ID interrupt. This bit is effective in LIN mode only. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The interrupt level is mapped to the INT1 line. <i>Write:</i> The interrupt level is mapped to the INT0 line.
12-10	Reserved	0	Read returns 0. Writes have no effect.
9	CLR RX INT LVL	0	Clear receiver interrupt. This bit is effective in LIN or SCI-compatible mode. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The interrupt level is mapped to the INT1 line. <i>Write:</i> The interrupt level is mapped to the INT0 line.
8	CLR TX INT LVL	0	Clear transmitter interrupt. This bit is effective in LIN or SCI-compatible mode. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The interrupt level is mapped to the INT1 line. <i>Write:</i> The interrupt level is mapped to the INT0 line.
7	CLR TOA3WUS INT LVL	0	Clear timeout after three wakeup signals interrupt. This bit is effective in LIN mode only. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The interrupt level is mapped to the INT1 line. <i>Write:</i> The interrupt level is mapped to the INT0 line.

**Table 24-18. SCI Clear Interrupt Level Register (SCICLEARINTLVL) Field Descriptions (continued)**

Bit	Field	Value	Description
6	CLR TOAWUS INT LVL	0	Clear timeout after wakeup signal interrupt. This bit is effective in LIN mode only. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The interrupt level is mapped to the INT1 line. <i>Write:</i> The interrupt level is mapped to the INT0 line.
5	Reserved	0	Read returns 0. Writes have no effect.
4	CLR TIMEOUT INT LVL	0	Clear timeout interrupt. This bit is effective in LIN mode only. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The interrupt level is mapped to the INT1 line. <i>Write:</i> The interrupt level is mapped to the INT0 line.
3-2	Reserved	0	Read returns 0. Writes have no effect.
1	CLR WAKEUP INT LVL	0	Clear wake-up interrupt. This bit is effective in LIN or SCI-compatible mode. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The interrupt level is mapped to the INT1 line. <i>Write:</i> The interrupt level is mapped to the INT0 line.
0	CLR BRKDT INT LVL	0	Clear break-detect interrupt. This bit is effective in SCI-compatible mode only. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The interrupt level is mapped to the INT1 line. <i>Write:</i> The interrupt level is mapped to the INT0 line.

### 24.13.8 SCI Flags Register (SCIFLR)

Figure 24-34 and Table 24-19 illustrate this register.

**Figure 24-34. SCI Flags Register (SCIFLR) [offset = 1Ch]**

31	30	29	28	27	26	25	24
BE	PBE	CE	ISFE	NRE	FE	OE	PE
R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/W-0	R/W-0	R/W-0
23							16
Reserved							
R-0							
15	14	13	12	11	10	9	8
Reserved	ID RX	ID TX	RX WAKE	TX EMPTY	TX WAKE	RX RDY	TX RDY
R-0	R/WL-0	R/WL-0	R/WC-0	R/W-1	R/WC-0	R/W-0	R/W-1
7	6	5	4	3	2	1	0
TOA3WUS	TOAWUS	Reserved	TIMEOUT	BUSY	IDLE	WAKE UP	BRKDT
R/WL-0	R/WL-0	R-0	R/WL-0	R/W-0	R-0	R/WL-0	R/WC-0

LEGEND: R/W = Read/Write; R = Read only; WC = Write in sci-compatible mode only; WL = Write in LIN mode only; -n = value after reset

**Table 24-19. SCI Flags Register (SCIFLR) Field Descriptions**

Bit	Field	Value	Description
31	BE	0 1	<p>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 <a href="#">Section 24.7.8</a> for more information. The bit error flag is cleared by any of the following:</p> <ul style="list-style-type: none"> <li>Setting of the SW nRST bit</li> <li>Setting of the RESET bit</li> <li>A system reset</li> <li>Writing a 1 to this bit</li> <li>On reception of a new synch break</li> <li>Reading the corresponding interrupt offset in SCIINTVECT0/1</li> </ul> <p><i>Read:</i> No error has been detected since this bit was last cleared. <i>Write:</i> Writing a 0 to this bit has no effect.</p> <p><i>Read:</i> An error has been detected since this bit was last cleared. <i>Write:</i> The bit is cleared to 0.</p>
30	PBE	0 1	<p>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 <a href="#">Section 24.7.8</a> for more information. The physical bus error flag is cleared by the following:</p> <ul style="list-style-type: none"> <li>Setting of the SW nRST bit</li> <li>Setting of the RESET bit</li> <li>A system reset</li> <li>Writing a 1 to this bit</li> <li>On reception of a new synch break</li> <li>Reading the corresponding interrupt offset in SCIINTVECT0/1</li> </ul> <p><b>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).</b></p> <p><i>Read:</i> No error has been detected since this bit was last cleared. <i>Write:</i> Writing a 0 to this bit has no effect.</p> <p><i>Read:</i> An error has been detected since this bit was last cleared. <i>Write:</i> The bit is cleared to 0.</p>



**Table 24-19. SCI Flags Register (SCIFLR) Field Descriptions (continued)**

Bit	Field	Value	Description
29	CE		<p>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 <a href="#">Section 24.7.8</a> 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:</p> <ul style="list-style-type: none"> <li>• Setting of the SW nRST bit</li> <li>• Setting of the RESET bit</li> <li>• A system reset</li> <li>• Writing a 1 to this bit</li> <li>• Reception of a new synch break</li> <li>• Reading the corresponding interrupt offset in SCIINTVECT0/1</li> </ul> <p>0 <i>Read:</i> No error has been detected since this bit was last cleared. <i>Write:</i> Writing a 0 to this bit has no effect.</p> <p>1 <i>Read:</i> An error has been detected since this bit was last cleared. <i>Write:</i> The bit is cleared to 0.</p>
28	ISFE		<p>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 <a href="#">Section 24.7.5.2</a> for more information. The inconsistent synch field error flag is cleared by the following:</p> <ul style="list-style-type: none"> <li>• Setting of the SW nRST bit</li> <li>• Setting of the RESET bit</li> <li>• A system reset</li> <li>• Writing a 1 to this bit</li> <li>• Reception of a new synch break</li> <li>• Reading the corresponding interrupt offset in SCIINTVECT0/1</li> </ul> <p>0 <i>Read:</i> No inconsistent synch field error has been detected. <i>Write:</i> Writing a 0 to this bit has no effect.</p> <p>1 <i>Read:</i> An inconsistent synch field error has been detected. <i>Write:</i> The bit is cleared to 0.</p>
27	NRE		<p>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 <a href="#">Section 24.7.7</a> for more information. The no-response error flag is cleared by the following:</p> <ul style="list-style-type: none"> <li>• Setting of the SW nRST bit</li> <li>• Setting of the RESET bit</li> <li>• A system reset</li> <li>• Writing a 1 to this bit</li> <li>• Reception of a new synch break</li> <li>• Reading the corresponding interrupt offset in SCIINTVECT0/1</li> </ul> <p>0 <i>Read:</i> No no-response error has been detected since the last clear. <i>Write:</i> Writing a 0 to this bit has no effect.</p> <p>1 <i>Read:</i> A no-response error has been detected. <i>Write:</i> The bit is cleared to 0.</p>

**Table 24-19. SCI Flags Register (SCIFLR) Field Descriptions (continued)**

Bit	Field	Value	Description
26	FE		<p>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 SCISSETINT. The framing error flag is cleared by the following:</p> <ul style="list-style-type: none"> <li>• Setting of the SW nRST bit</li> <li>• Setting of the RESET bit</li> <li>• A system reset</li> <li>• Writing a 1 to this bit</li> <li>• Reading the corresponding interrupt offset in SCIINTVECT0/1</li> <li>• Reception of a new character/frame, depending on whether the module is in SCI compatible or LIN mode</li> </ul> <p>In multibuffer mode the frame is defined in the SCIFORMAT register.</p> <p>0 <i>Read:</i> No framing error has been detected since the last clear. <i>Write:</i> Writing a 0 to this bit has no effect.</p> <p>1 <i>Read:</i> A framing error has been detected since the last clear. <i>Write:</i> The bit is cleared to 0.</p>
25	OE		<p>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:</p> <ul style="list-style-type: none"> <li>• Setting of the SW nRST bit</li> <li>• Setting of the RESET bit</li> <li>• A system reset</li> <li>• Writing a 1 to this bit</li> <li>• Reading the corresponding interrupt offset in SCIINTVECT0/1</li> </ul> <p>0 <i>Read:</i> No overrun error has been detected since the last clear. <i>Write:</i> Writing a 0 to this bit has no effect.</p> <p>1 <i>Read:</i> An overrun error has been detected. <i>Write:</i> The bit is cleared to 0.</p>
24	PE		<p>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:</p> <ul style="list-style-type: none"> <li>• Setting of the SW nRST bit</li> <li>• Setting of the RESET bit</li> <li>• A system reset</li> <li>• Writing a 1 to this bit</li> <li>• Reception of a new character or frame, depending on whether the module is in SCI compatible or LIN mode, respectively.</li> <li>• Reading the corresponding interrupt offset in SCIINTVECT0/1</li> </ul> <p>0 <i>Read:</i> No parity error has been detected since the last clear. <i>Write:</i> Writing a 0 to this bit has no effect.</p> <p>1 <i>Read:</i> A parity error has been detected. <i>Write:</i> The bit is cleared to 0.</p>
23-15	Reserved	0	Read returns 0. Writes have no effect.

**Table 24-19. SCI Flags Register (SCIFLR) Field Descriptions (continued)**

Bit	Field	Value	Description
14	ID RX FLAG		<p>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 <a href="#">Section 24.7.9</a> 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:</p> <ul style="list-style-type: none"> <li>• Setting of the SW nRST bit</li> <li>• Setting of the RESET bit</li> <li>• A system reset</li> <li>• Writing a 1 to this bit</li> <li>• Reading the LINID register</li> <li>• Reception of a new synch break</li> <li>• Reading the corresponding interrupt offset in SCIINTVECT0/1</li> </ul> <p>0 <i>Read:</i> No valid ID has been received since the last clear. <i>Write:</i> Writing a 0 to this bit has no effect.</p> <p>1 <i>Read:</i> A valid ID RX has been received in LINID[23:16] on an RX match. <i>Write:</i> The bit is cleared to 0.</p>
13	ID TX FLAG		<p>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 <a href="#">Section 24.7.9</a> 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:</p> <ul style="list-style-type: none"> <li>• Setting the SW nRST bit</li> <li>• Setting of the RESET bit</li> <li>• A system reset</li> <li>• Writing a 1 to this bit</li> <li>• Reading the LINID register</li> <li>• Receiving a new synch break</li> <li>• Reading the corresponding interrupt offset in SCIINTVECT0/1</li> </ul> <p>0 <i>Read:</i> No valid ID has been received since the last clear. <i>Write:</i> Writing a 0 to this bit has no effect.</p> <p>1 <i>Read:</i> A valid ID TX has been received in LINID[23:16] on an TX match. <i>Write:</i> The bit is cleared to 0.</p>
12	RXWAKE		<p>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:</p> <ul style="list-style-type: none"> <li>• Setting of the SW nRST bit</li> <li>• Setting of the RESET bit</li> <li>• A system reset</li> <li>• Upon receipt of a data frame.</li> </ul> <p>0 The data in SCIRD is not an address.</p> <p>1 The data in SCIRD is an address.</p>
11	TX EMPTY		<p>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.</p> <p><b>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.</b></p> <p><i>SCI mode or LIN non-multibuffer mode</i></p> <p>0 Transmitter buffer or shift register (or both) are loaded with data.</p> <p>1 Transmitter buffer and shift registers are both empty.</p> <p><i>In LIN mode using multibuffer mode</i></p> <p>0 Multibuffer or shift register (or all) are loaded with data</p> <p>1 Multibuffer and shift registers are all empty.</p>

**Table 24-19. SCI Flags Register (SCIFLR) Field Descriptions (continued)**

Bit	Field	Value	Description
10	TXWAKE		<p>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.</p> <p><b>Note: TXWAKE is not cleared by the SW nRESET bit.</b></p> <p><i>Address-bit mode</i></p> <p>0 Frame to be transmitted will be data (address bit = 0). 1 Frame to be transmitted will be an address (address bit = 1).</p> <p><i>Idle-line mode</i></p> <p>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).</p>
9	RXRDY		<p>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 <i>LIN mode</i>, RXRDY is set once a valid frame is received in multibuffer mode, a valid frame being a message frame received with no errors. In <i>non-multibuffer mode</i>, 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 (SCISSETINT[9]); RXRDY is cleared by the following:</p> <ul style="list-style-type: none"> <li>• Setting of the SW nRST bit</li> <li>• Setting of the RESET bit</li> <li>• A system reset</li> <li>• Writing a 1 to this bit</li> <li>• Reading the SCIRD register in compatibility mode</li> <li>• Reading the last data byte RDy of the response in LIN mode</li> </ul> <p><b>Note: The RXRDY flag cannot be cleared by reading the corresponding interrupt offset in the SCIINTVECT0/1 register.</b></p> <p>0 <i>Read:</i> No new data is in SCIRD. <i>Write:</i> Writing a 0 to this bit has no effect.</p> <p>1 <i>Read:</i> New data is ready to be read from SCIRD. <i>Write:</i> The bit is cleared to 0.</p>
8	TXRDY		<p>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.</p> <p><i>In SCI</i>, writing data to SCITD automatically clears this bit. <i>In LIN mode</i>, 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.</p> <p><b>Note:</b></p> <ol style="list-style-type: none"> <li>1) TXRDY is also set to 1 either by setting of the RESET bit, enabling SW nRST or by a system reset.</li> <li>2) The TXRDY flag cannot be cleared by reading the corresponding interrupt offset in the SCIINTVECT0/1 register.</li> <li>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.</li> </ol> <p><i>SCI mode</i></p> <p>0 SCITD is full. 1 SCITD is ready to receive the next character.</p> <p><i>LIN mode</i></p> <p>0 The multibuffers are full. 1 The multibuffers are ready to receive the next character(s).</p> <p>For more information on transmit interrupt handling, see the SCI document for compatibility mode and <a href="#">Section 24.7.9</a> for LIN mode.</p>

**Table 24-19. SCI Flags Register (SCIFLR) Field Descriptions (continued)**

Bit	Field	Value	Description
7	TOA3WUS	0 1	<p>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:</p> <ul style="list-style-type: none"> <li>Setting of the SW nRST bit</li> <li>Setting of the RESET bit</li> <li>A system reset</li> <li>Writing a 1 to this bit</li> <li>Reading the corresponding interrupt offset in SCIINTVECT0/1</li> </ul> <p>See <a href="#">Section 24.11.3</a> for more information.</p> <p><b>0</b> <i>Read:</i> No timeout occurred after three wakeup signals. <i>Write:</i> Writing a 0 to this bit has no effect.</p> <p><b>1</b> <i>Read:</i> Timeout occurred after three wakeup signals and 1.5 seconds time. <i>Write:</i> The bit is cleared to 0.</p>
6	TOAWUS	0 1	<p>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:</p> <ul style="list-style-type: none"> <li>Setting the SW nRST bit</li> <li>Setting of the RESET bit</li> <li>A system reset occurring</li> <li>Writing a 1 to this bit</li> <li>Reading the corresponding interrupt offset in SCIINTVECT0/1</li> </ul> <p>See <a href="#">Section 24.11.3s</a> for more information.</p> <p><b>0</b> <i>Read:</i> No timeout occurred after one wakeup signal (150ms). <i>Write:</i> Writing a 0 to this bit has no effect.</p> <p><b>1</b> <i>Read:</i> Timeout occurred after one wakeup signal. <i>Write:</i> The bit is cleared to 0.</p>
5	Reserved	0	Read returns 0. Writes have no effect.
4	TIMEOUT	0 1	<p>LIN bus idle timeout flag. This bit is effective in LIN mode only. This bit is set earliest after at least four seconds of bus inactivity. Bus inactivity is defined as no transactions between recessive and dominant (and vice versa). This bit is cleared by the following:</p> <ul style="list-style-type: none"> <li>Setting the SW nRST bit</li> <li>Setting of the RESET bit</li> <li>A system reset occurring</li> <li>Writing a 1 to this bit</li> <li>Reading the corresponding interrupt offset in SCIINTVECT0/1</li> </ul> <p>See <a href="#">Section 24.7.7</a> for more information.</p> <p><b>0</b> <i>Read:</i> No bus idle has been detected since this bit was last cleared. <i>Write:</i> Writing a 0 to this bit has no effect.</p> <p><b>1</b> <i>Read:</i> A LIN bus idle has been detected. <i>Write:</i> The bit is cleared to 0.</p>
3	BUSY	0 1	<p>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 lowpower 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:</p> <ul style="list-style-type: none"> <li>Setting the SW nRST bit</li> <li>Setting of the RESET bit</li> <li>A system reset occurring</li> </ul> <p><b>0</b> The receiver is not currently receiving a frame.</p> <p><b>1</b> The receiver is currently receiving a frame.</p>

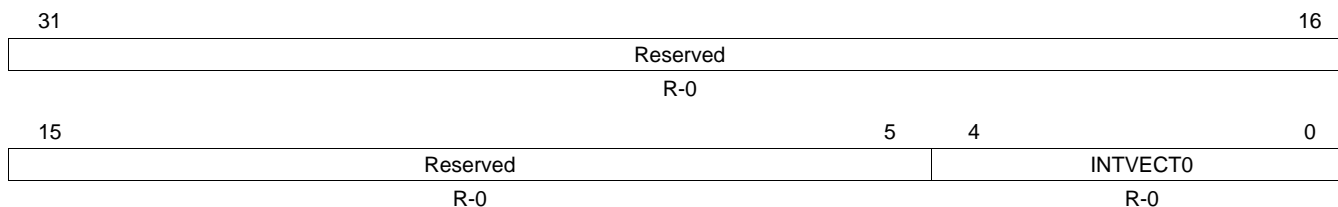
**Table 24-19. SCI Flags Register (SCIFLR) Field Descriptions (continued)**

Bit	Field	Value	Description
2	IDLE	0 1	<p>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:</p> <ul style="list-style-type: none"> <li>• A system reset</li> <li>• An SCI software reset</li> <li>• A power down</li> <li>• The RX pin is configured as a general I/O pin</li> </ul> <p>The idle period has been detected; the SCI is ready to receive.</p> <p>The idle period has not been detected; the SCI will not receive any data.</p>
1	WAKEUP	0 1	<p>Wakeup flag. This bit is effective in LIN mode only. This bit is set by the SCI/LIN when receiver or transmitter activity has taken the module out of power-down mode. An interrupt is generated if the SET WAKEUP INT bit (SCISSETINT[2]) is set. It is cleared by the following:</p> <ul style="list-style-type: none"> <li>• Setting of the SW nRST bit</li> <li>• Setting of the RESET bit</li> <li>• A system reset</li> <li>• Writing a 1 to this bit</li> <li>• Reading the corresponding interrupt offset in SCIINTVECT0/1</li> </ul> <p>For compatibility mode, see the SCI document for more information on low-power mode.</p> <p><i>Read:</i> The module will not wake up from power-down mode. <i>Write:</i> Writing a 0 to this bit has no effect.</p> <p><i>Read:</i> Wake up from power-down mode. <i>Write:</i> The bit is cleared to 0.</p>
0	BRKDT	0 1	<p>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:</p> <ul style="list-style-type: none"> <li>• Setting of the SW nRST bit</li> <li>• Setting of the RESET bit</li> <li>• A system reset</li> <li>• Writing a 1 to this bit</li> <li>• Reading the corresponding interrupt offset in SCIINTVECT0/1</li> </ul> <p><i>Read:</i> No break condition has been detected since the last clear. <i>Write:</i> Writing a 0 to this bit has no effect.</p> <p><i>Read:</i> A break condition has been detected. <i>Write:</i> The bit is cleared to 0.</p>

### 24.13.9 SCI Interrupt Vector Offset 0 (SCIINTVECT0)

Figure 24-35 and Table 24-20 illustrate this register.

**Figure 24-35. SCI Interrupt Vector Offset 0 (SCIINTVECT0) [offset = 20h]**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

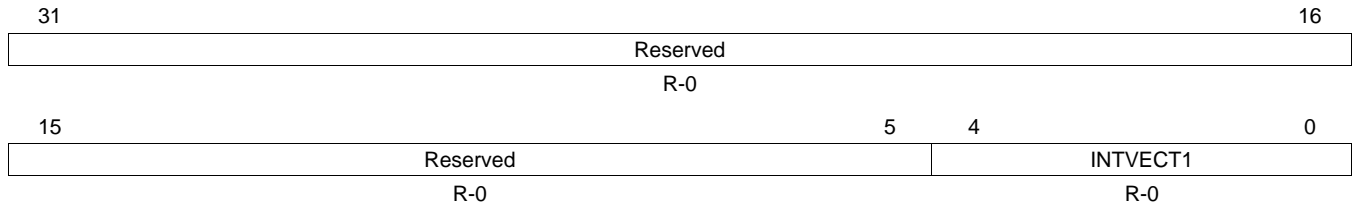
**Table 24-20. SCI Interrupt Vector Offset 0 (SCIINTVECT0) Field Descriptions**

Bit	Field	Value	Description
31-5	Reserved	0	Read returns 0. Writes have no effect.
4-0	INVECT0	0-1Fh	Interrupt vector offset for INT0. This register indicates the offset for interrupt line INT0. A read to this register updates its value to the next highest priority pending interrupt in SCIFLR and clears the flag in SCIFLR corresponding to the offset that was read. See <a href="#">Table 24-4</a> for a list of the interrupts.  <b>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).</b>

**24.13.10 SCI Interrupt Vector Offset 1 (SCIINTVECT1)**

Figure 24-36 and Table 24-21 illustrate this register.

**Figure 24-36. SCI Interrupt Vector Offset 1 (SCIINTVECT1) [offset = 24h]**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 24-21. SCI Interrupt Vector Offset 1 (SCIINTVECT1) Field Descriptions**

Bit	Field	Value	Description
31-5	Reserved	0	Read returns 0. Writes have no effect.
4-0	INTECT1	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 24-4 for list of interrupts.  <b>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).</b>



### 24.13.11 SCI Format Control Register (SCIFORMAT)

Figure 24-37 and Table 24-22 illustrate this register.

**Figure 24-37. SCI Format Control Register (SCIFORMAT) [offset = 28h]**

31	19	18	16
Reserved		LENGTH	
R-0		R/W-0	
15	3	2	0
Reserved		CHAR	
R-0		R/WC-0	

LEGEND: R/W = Read/Write; R = Read only; WC = Write in SCI-compatible mode only; -n = value after reset

**Table 24-22. SCI Format Control Register (SCIFORMAT) Field Descriptions**

Bit	Field	Value	Description
31-19	Reserved	0	Read returns 0. Writes have no effect.
18-16	LENGTH	0-3h	<p>Frame length control bits. In <i>LIN mode</i>, these bits indicate the number of bytes in the response field from 1 to 8 bytes. In <i>buffered SCI mode</i>, these bits indicate the number of characters, with the number of bits per character specified in CHAR (SCIFORMAT[2:0]).</p> <p>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.</p>
		0	The response field has 1 byte/character.
		1h	The response field has 2 bytes/characters.
		2h	The response field has 3 bytes/characters.
		3h	The response field has 4 bytes/characters.
		4h	The response field has 5 bytes/characters.
		5h	The response field has 6 bytes/characters.
		6h	The response field has 7 bytes/characters.
		7h	The response field has 8 bytes/characters.
15-3	Reserved	0	Read returns 0. Writes have no effect.
2-0	CHAR	0-7h	<p>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.</p> <p><b>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.</b></p> <p><b>Data written to the SCITD should be right justified but does not need to be padded with leading zeros.</b></p>
		0	The character is 1 bit long.
		1h	The character is 2 bits long.
		2h	The character is 3 bits long.
		3h	The character is 4 bits long.
		4h	The character is 5 bits long.
		5h	The character is 6 bits long.
		6h	The character is 7 bits long.
		7h	The character is 8 bits long.

**24.13.12 Baud Rate Selection Register (BRS)**

This section describes the baud rate selection register. [Figure 24-38](#) and [Table 24-23](#) illustrate this register.

**Figure 24-38. Baud Rate Selection Register (BRS) [offset = 2Ch]**

31	30	28	27	24	23	16
Rsvd.	U			M	PRESCALER P	
R-0	R/W-0			R/W-0	R/W-0	
15						0
PRESCALER P						
R/W-0						

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 24-23. Baud Rate Selection Register (BRS) Field Descriptions**

Bit	Field	Value	Description
31	Reserved	0	Read returns 0. Writes have no effect.
30-28	U	0-2h	SCI/LIN super fractional divider selection. These bits are effective in LIN or SCI asynchronous mode. These bits are an additional fractional part for the baud rate specification. These bits allow a super-fine tuning of the fractional baud rate with seven more intermediate values for each of the M fractional divider values. See <a href="#">Section 24.7.4.1</a> for more details.
27-24	M	0-3h	SCI/LIN 4-bit fractional divider selection. These bits are effective in LIN or SCI asynchronous mode. These bits are used to select a baud rate for the SCI/LIN module, and they are a fractional part for the baud rate specification. The M divider allows fine-tuning of the baud rate over the P prescaler with 15 additional intermediate values for each of the P integer values. See <a href="#">Section 24.7.4.1</a> for more details.
23-0	PRESCALER P	0-FF FFFFh	<p>These bits are used to select a baud rate for the SCI/LIN module. These bits are effective in LIN mode and SCI compatibility..</p> <p>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].</p> <p><b>NOTE: In LIN mode, ONLY the asynchronous mode and baudrate values are used.</b></p> <p>The baud rate can be calculated using the following formulas:</p> $\text{Asynchronous baud value} = \frac{\text{VCLK Frequency}}{16 \left( P + 1 + \frac{M}{16} \right)} \quad (47)$ $\text{Isosynchronous baud value} = \frac{\text{VCLK Frequency}}{P + 1} \quad (48)$ <p>For P = 0,</p> $\text{Asynchronous baud value} = \frac{\text{VCLK Frequency}}{32} \quad (49)$ $\text{Isosynchronous baud value} = \frac{\text{VCLK Frequency}}{2} \quad (50)$ <p><a href="#">Table 24-24</a> contains comparative baud values for different P values, with VCLK = 50 MHz, for asynchronous mode.</p>

**Table 24-24. Comparative Baud Values for Different P Values, Asynchronous Mode<sup>(1)(2)</sup>**

24-Bit Register Value		Baud Selected		PercentError
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

(1) VCLK = 50 MHz

(2) Values are in decimal except for column 2.

**Table 24-24. Comparative Baud Values for Different P Values, Asynchronous Mode<sup>(1)(2)</sup> (continued)**

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

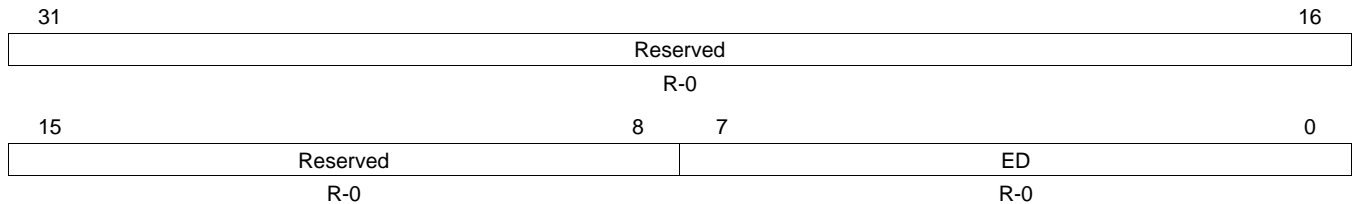
**24.13.13 SCI Data Buffers (SCIED, SCIRD, SCITD)**

The SCI has three addressable registers in which transmit and receive data is stored. These three registers are available in SCI mode only.

**24.13.13.1 Receiver Emulation Data Buffer (SCIED)**

The SCIED register is addressed at a location different from SCIRD, but is physically the same register. [Figure 24-39](#) and [Table 24-25](#) illustrate this register.

**Figure 24-39. Receiver Emulation Data Buffer (SCIED) [offset = 30h]**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 24-25. Receiver Emulation Data Buffer (SCIED) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Read returns 0. Writes have no effect.
7-0	ED	0-FFh	Emulator data. This bit is effective in SCI-compatible mode only. Reading SCIED[7:0] does not clear the RXRDY flag, unlike reading SCIRD. This register should be used only by an emulator that must continually read the data buffer without affecting the RXRDY flag.

### 24.13.13.2 Receiver Data Buffer (SCIRD)

This register provides a location for the receiver data. [Figure 24-40](#) and [Table 24-26](#) illustrate this register.

**Figure 24-40. Receiver Data Buffer (SCIRD) [offset = 34h]**

31	Reserved			16
	R-0			
15	8	7		0
Reserved		RD		
R-0		R-0		

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

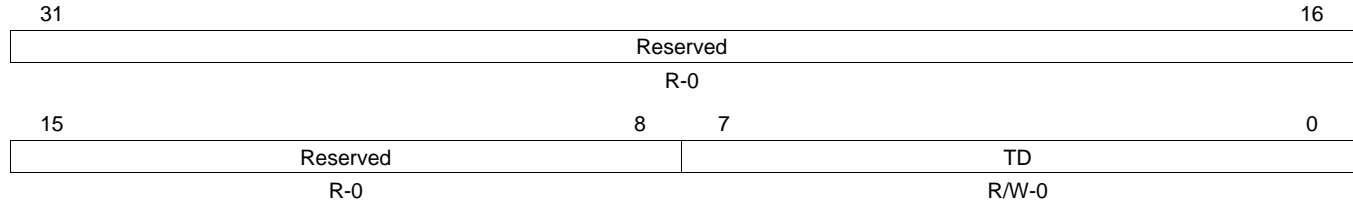
**Table 24-26. Receiver Data Buffer (SCIRD) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Read returns 0. Writes have no effect.
7-0	RD	0-FFh	Receiver data. This bit is effective in SCI-compatible mode only. When a frame has been completely received, the data in the frame is transferred from the receiver shift register SCIRXSHF to this register. As this transfer occurs, the RXRDY flag is set and a receive interrupt is generated if SET RX INT is set. <b>Note: When the data is read from SCIRD, the RXRDY flag is automatically cleared.</b>

**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.

**24.13.13.3 Transmit Data Buffer Register (SCITD)**

Data to be transmitted is written to the SCITD register. [Figure 24-41](#) and [Table 24-27](#) illustrate this register.

**Figure 24-41. Transmit Data Buffer Register (SCITD) [offset = 38h]**


LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 24-27. Transmit Data Buffer Register (SCITD) Field Descriptions**

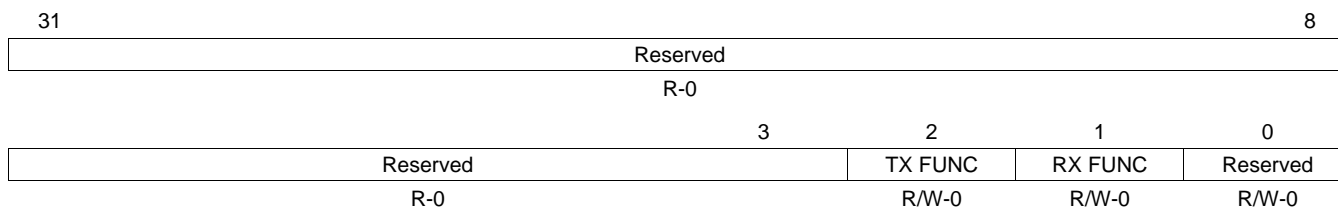
Bit	Field	Value	Description
31-8	Reserved	0	Read returns 0. Writes have no effect.
7-0	TD	0-FFh	Transmit data. This bit is effective in SCI-compatible mode only. Data to be transmitted is written to the SCITD register. The transfer of data from this register to the transmit shift register SCITXSHF sets the TXRDY flag, which indicates that SCITD is ready to be loaded with another byte of data.  <b>Note: If TX INT ENA is set, this data transfer also causes an interrupt.</b>

**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.

### 24.13.14 SCI Pin I/O Control Register 0 (SCIPIO0)

Figure 24-42 and Table 24-28 illustrate this register.

**Figure 24-42. SCI Pin I/O Control Register 0 (SCIPIO0) [offset = 3Ch]**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 24-28. SCI Pin I/O Control Register 0 (SCIPIO0) Field Descriptions**

Bit	Field	Value	Description
31-3	Reserved	0	Read returns 0. Writes have no effect.
2	TX FUNC	0	Transfer function. This bit is effective in LIN or SCI mode. This bit defines the function of pin LINTX. LINTX is a general-purpose digital I/O pin.
		1	LINTX is the SCI/LIN transmit pin.
1	RX FUNC	0	Receive function. This bit is effective in LIN or SCI mode. This bit defines the function of pin LINRX. LINRX is a general-purpose digital I/O pin.
		1	LINRX is the SCI/LIN receive pin.
0	Reserved		Writes have no effect.

### 24.13.15 SCI Pin I/O Control Register 1 (SCIPIO1)

Figure 24-43 and Table 24-29 illustrate this register.

**Figure 24-43. SCI Pin I/O Control Register 1 (SCIPIO1) [offset = 40h]**

31	Reserved	8
	R-0	
	3	2
	1	0
	Reserved	TX DIR
	R-0	R/W-0
		R/W-0
		Reserved
		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 24-29. SCI Pin I/O Control Register 1 (SCIPIO1) Field Descriptions**

Bit	Field	Value	Description
31-3	Reserved	0	Read returns 0. Writes have no effect.
2	TX DIR	0	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 24-30 for the LINTX pin control with this bit and others.
		1	LINTX is a general-purpose input pin.
		1	LINTX is a general-purpose output pin.
1	RX DIR	0	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 24-31 for the LINRX pin control with this bit and others.
		1	LINRX is a general-purpose input pin.
		1	LINRX is a general-purpose output pin.
0	Reserved		Writes have no effect.

**Table 24-30. LINTX Pin Control**

Function	TX IN <sup>(1)</sup>	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

<sup>(1)</sup> TX IN is a read-only bit. Its value always reflects the level of the SCITX pin.

**Table 24-31. LINRX Pin Control**

Function	RX IN <sup>(1)</sup>	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

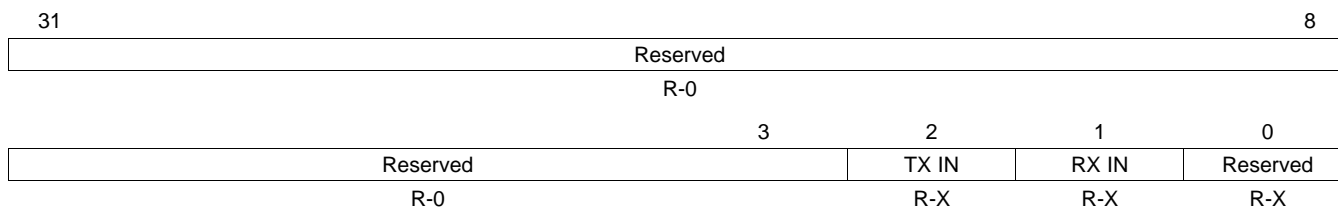
<sup>(1)</sup> RX IN is a read-only bit. Its value always reflects the level of the SCIRX pin.



### 24.13.16 SCI Pin I/O Control Register 2 (SCIPIO2)

Figure 24-44 and Table 24-32 illustrate this register.

**Figure 24-44. SCI Pin I/O Control Register 2 (SCIPIO2) [offset = 44h]**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset; -x = Indeterminate

**Table 24-32. SCI Pin I/O Control Register 2 (SCIPIO2) Field Descriptions**

Bit	Field	Value	Description
31-3	Reserved	0	Read returns 0. Writes have no effect.
2	TX IN	0	Transmit pin in. This bit is effective in LIN or SCI mode. This bit contains the current value on the LINTX pin. The LINTX pin is at logic low (0).
		1	The LINTX pin is at logic high (1).
1	RX IN	0	Receive pin in. This bit is effective in LIN or SCI mode. This bit contains the current value on the LINRX pin. The LINRX pin is at logic low (0).
		1	The LINRX pin is at logic high (1).
0	Reserved		Writes have no effect.

**24.13.17 SCI Pin I/O Control Register 3 (SCIPIO3)**

Figure 24-45 and Table 24-33 illustrate this register.

**Figure 24-45. SCI Pin I/O Control Register 3 (SCIPIO3) [offset = 48h]**

31	Reserved		8
	R-0		
	3	2	1
	Reserved	TX OUT	RX OUT
	R-0	R/W-0	R/W-0
		0	Reserved
		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 24-33. SCI Pin I/O Control Register 3 (SCIPIO3) Field Descriptions**

Bit	Field	Value	Description
31-3	Reserved	0	Read returns 0. Writes have no effect.
2	TX OUT	0 1	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: <ul style="list-style-type: none"> <li>• TX FUNC = 0 (LINTX pin is a general-purpose I/O.)</li> <li>• TX DIR = 1 (LINTX pin is a general-purpose output.)</li> </ul> See Table 24-30 for an explanation of this bit's effect in combination with other bits. 0 The output on the LINTX is at logic low (0). 1 The output on the LINTX pin is at logic high (1). (Output voltage is $V_{OH}$ or higher if TXPDR = 0 and output is in high impedance state if TXPDR = 1)
1	RX OUT	0 1	Receive pin out. This bit is effective in LIN or SCI mode. This bit specifies the logic to be output on pin LINRX if the following conditions are met: <ul style="list-style-type: none"> <li>• RX FUNC = 0 (LINRX pin is a general-purpose I/O.)</li> <li>• RX DIR = 1 (LINRX pin is a general-purpose output.)</li> </ul> See Table 24-31 for an explanation of this bit's effect in combination with the other bits. 0 The output on the LINRX pin is at logic low (0). 1 The output on the LINRX pin is at logic high (1). (Output voltage is $V_{OH}$ or higher if RXPDR = 0, and output is in high impedance state if RXPDR = 1)
0	Reserved		Writes have no effect.

### 24.13.18 SCI Pin I/O Control Register 4 (SCIPIO4)

Figure 24-46 and Table 24-34 illustrate this register.

**Figure 24-46. SCI Pin I/O Control Register 4 (SCIPIO4) [offset = 4Ch]**

31	Reserved				8	
R-0						
			3	2	1	0
Reserved			TX SET	RX SET	Reserved	
R-0			R/W-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 24-34. SCI Pin I/O Control Register 4 (SCIPIO4) Field Descriptions**

Bit	Field	Value	Description
31-3	Reserved	0	Read returns 0. Writes have no effect.
2	TX SET	0	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: <ul style="list-style-type: none"> <li>TX FUNC = 0 (LINTX pin is a general-purpose I/O.)</li> <li>TX DIR = 1 (LINTX pin is a general-purpose output.)</li> </ul> See Table 24-30 for an explanation of this bit's effect in combination with other bits.
		1	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	0	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: <ul style="list-style-type: none"> <li>RX FUNC = 0 (LINRX pin is a general-purpose I/O.)</li> <li>RX DIR = 1 (LINRX pin is a general-purpose output.)</li> </ul> See Table 24-31 for an explanation of this bit's effect in combination with the other bits.
		1	Read: The output on LINRX is at logic low (0). Write: Writing a 0 to this bit has no effect.
		1	Read and write: The output on LINRX is at logic high (1).
0	Reserved		Writes have no effect.

### 24.13.19 SCI Pin I/O Control Register 5 (SCIPIO5)

Figure 24-47 and Table 24-35 illustrate this register.

**Figure 24-47. SCI Pin I/O Control Register 5 (SCIPIO5) [offset = 50h]**

31	Reserved				8
	R-0				
		3	2	1	0
	Reserved	TX CLR	RX CLR	Reserved	
	R-0	R/W-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 24-35. SCI Pin I/O Control Register 5 (SCIPIO5) Field Descriptions**

Bit	Field	Value	Description
31-3	Reserved	0	Read returns 0. Writes have no effect.
2	TX CLR	0 1	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: <ul style="list-style-type: none"> <li>• TX FUNC = 0 (LINTX pin is a general-purpose I/O.)</li> <li>• TX DIR = 1 (LINTX pin is a general-purpose output.)</li> </ul> <i>Read:</i> The output on LINTX is at logic low (0). <i>Write:</i> Writing a 0 to this bit has no effect.  <i>Read:</i> The output on LINTX is at logic high (1). <i>Write:</i> The output on LINTX is at logic low (0).
1	RX CLR	0 1	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: <ul style="list-style-type: none"> <li>• RX FUNC = 0 (LINRX pin is a general-purpose I/O.)</li> <li>• RX DIR = 1 (LINRX pin is a general-purpose output.)</li> </ul> <i>Read:</i> The output on LINRX is at logic low (0). <i>Write:</i> Writing a 0 to this bit has no effect.  <i>Read:</i> The output on LINRX is at logic high (1). <i>Write:</i> The output on LINRX is at logic low (0).
0	Reserved		Writes have no effect.

### 24.13.20 SCI Pin I/O Control Register 6 (SCIPIO6)

Figure 24-48 and Table 24-36 illustrate this register.

**Figure 24-48. SCI Pin I/O Control Register 6 (SCIPIO6) [offset = 54h]**

31	Reserved				8	
R-0						
			3	2	1	0
Reserved			TX FUNC	RX FUNC	Reserved	
R-0			R/W-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 24-36. SCI Pin I/O Control Register 6 (SCIPIO6) Field Descriptions**

Bit	Field	Value	Description
31-3	Reserved	0	Read returns 0. Writes have no effect.
2	TX PDR	0	Transmit pin open drain enable. This bit is effective in LIN or SCI mode. This bit enables open-drain capability in the output pin LINTX if the following conditions are met: <ul style="list-style-type: none"> <li>TX FUNC = 0 (LINTX pin is a general-purpose I/O.)</li> <li>TX DIR = 1 (LINTX pin is a general-purpose output.)</li> </ul>
		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	0	Receive pin open drain enable. This bit is effective in LIN or SCI mode. This bit enables open-drain capability in the output pin LINRX if the following conditions are met: <ul style="list-style-type: none"> <li>RX FUNC = 0 (LINRX pin is a general-purpose I/O.)</li> <li>RX DIR = 1 (LINRX pin is a general-purpose output.)</li> </ul>
		0	Open drain functionality is disabled; the output voltage is $V_{OL}$ or lower if RXOUT =0 and $V_{OH}$ or higher if RXOUT =1.
		1	Open drain functionality is enabled; the output voltage is $V_{OL}$ or lower if RXOUT =0 and high impedance if RXOUT =1.
0	Reserved		Writes have no effect.

**24.13.21 SCI Pin I/O Control Register 7 (SCIPIO7)**

Figure 24-49 and Table 24-37 illustrate this register.

**Figure 24-49. SCI Pin I/O Control Register 7 (SCIPIO7) [offset = 58h]**

31	Reserved	8
	R-0	
	3	2
	1	0
	Reserved	TX PD
	Reserved	RX PD
	Reserved	Reserved
	R-0	R/W-n
		R/W-n
		R/W-n

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset, Refer to the Terminal Functions in the device datasheet for default pin settings.

**Table 24-37. SCI Pin I/O Control Register 7 (SCIPIO7) Field Descriptions**

Bit	Field	Value	Description
31-3	Reserved	0	Read returns 0. Writes have no effect.
2	TX PD	0	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. The pull control on the LINTX pin is enabled.
		1	The pull control on the LINTX pin is disabled.
1	RX PD	0	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. Pull control on the LINRX pin is enabled.
		1	Pull control on the LINRX pin is disabled.
0	Reserved		Writes have no effect.

### 24.13.22 SCI Pin I/O Control Register 8 (SCIPIO8)

Figure 24-50 and Table 24-38 illustrate this register.

**Figure 24-50. SCI Pin I/O Control Register 8 (SCIPIO8) [offset = 5Ch]**

31	Reserved				8	
R-0						
			3	2	1	0
Reserved			TX PSL	RX PSL	Reserved	
R-0			R/W-n	R/W-n	R/W-n	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset, Refer to the Terminal Functions in the device datasheet for default pin settings.

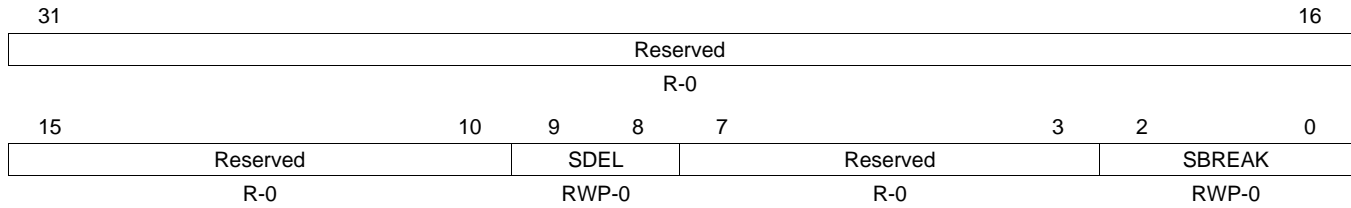
**Table 24-38. SCI Pin I/O Control Register 8 (SCIPIO8) Field Descriptions**

Bit	Field	Value	Description
31-3	Reserved	0	Read returns 0. Writes have no effect.
2	TX PSL	0	The LINTX pin is a pull down.
		1	The LINTX pin is a pull up.
1	RX PSL	0	The LINRX pin is a pull down.
		1	The LINRX pin is a pull up.
0	Reserved		Writes have no effect.

### 24.13.23 LIN Compare Register (LINCOMPARE)

Figure 24-51 and Table 24-39 illustrate this register.

**Figure 24-51. LIN Compare Register (LINCOMPARE) [offset = 60h]**



LEGEND: R/W = Read/Write; R = Read only; RWP = Read/Write in privileged mode only; --n = value after reset

**Table 24-39. LIN Compare Register (LINCOMPARE) Field Descriptions**

Bit	Field	Value	Description
31-19	Reserved	0	Read returns 0. Writes have no effect.
18-16	SDEL	0 1h 2h 3h	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 00.  The formula to program the value (in $T_{bits}$ ) for the synchronization delimiter is:  $T_{SDEL} = (SDEL + 1) T_{bit}$  The synch delimiter has 1 $T_{bit}$ . The synch delimiter has 2 $T_{bit}$ . The synch delimiter has 3 $T_{bit}$ . The synch delimiter has 4 $T_{bit}$ .
15-3	Reserved	0	Read returns 0. Writes have no effect.
2-0	SBREAK	0 1h 2h 3h 4h 5h 6h 7h	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:  $T_{SYNBRK} = 13T_{bit} + SBREAK \times T_{bit}$  The synch break has no additional $T_{bit}$ . The synch break has 1 additional $T_{bit}$ . The synch break has 2 additional $T_{bit}$ . The synch break has 3 additional $T_{bit}$ . The synch break has 4 additional $T_{bit}$ . The synch break has 5 additional $T_{bit}$ . The synch break has 6 additional $T_{bit}$ . The synch break has 7 additional $T_{bit}$ .



### 24.13.24 LIN Receive Buffer 0 Register (LINRD0)

Figure 24-52 and Table 24-40 illustrate this register.

**Figure 24-52. LIN Receive Buffer 0 Register (LINRD0) [offset = 64h]**

31	24	23	16
RD0		RD1	
R-0		R-0	
15	8	7	0
RD2		RD3	
R-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 24-40. LIN Receive Buffer 0 Register (LINRD0) Field Descriptions**

Bit	Field	Value	Description
31-24	RD0	0-FFh	Receive buffer 0. Byte 0 of the response data byte. Each response data-byte that is received in the SCIRXSHFT register is transferred to the corresponding RDy bit field according to the number of bytes received. A read of this byte clears the RXDY byte. <b>Note: RD&lt;x-1&gt; is equivalent to data byte &lt;x&gt; of the LIN frame.</b>
23-16	RD1	0-FFh	Receive buffer 1. Byte 1 of the response data byte. Each response data-byte that is received in the SCIRXSHFT register is transferred to the corresponding RDy register according to the number of bytes received.
15-8	RD2	0-FFh	Receive buffer 2. Byte 2 of the response data byte. Each response data-byte that is received in the SCIRXSHFT register is transferred to the corresponding RDy register according to the number of bytes received.
7-0	RD3	0-FFh	Receive buffer 3. Byte 3 of the response data byte. Each response data-byte that is received in the SCIRXSHFT register is transferred to the corresponding RDy register according to the number of bytes received.

### 24.13.25 LIN Receive Buffer 1 Register (LINRD1)

Figure 24-53 and Table 24-41 illustrate this register.

**Figure 24-53. LIN Receive Buffer 1 Register (RD1) [offset = 68h]**

31	24	23	16
RD4		RD5	
R-0		R-0	
15	8	7	0
RD6		RD7	
R-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 24-41. LIN Receive Buffer 1 Register (RD1) Field Descriptions**

Bit	Field	Value	Description
31-24	RD4	0-FFh	Receive buffer 4. Byte 4 of the response data byte. Each response data-byte that is received in the SCIRXSHFT register is transferred to the corresponding register according to the number of bytes received. <b>Note: RD&lt;x-1&gt; is equivalent to data byte &lt;x&gt; of the LIN frame.</b>
23-16	RD5	0-FFh	Receive buffer 5. Byte 5 of the response data byte. Each response data-byte that is received in the SCIRXSHFT register is transferred to the corresponding register according to the number of bytes received.
15-8	RD6	0-FFh	Receive buffer 6. Byte 6 of the response data byte. Each response data-byte that is received in the SCIRXSHFT register is transferred to the corresponding register according to the number of bytes received.
7-0	RD7	0-FFh	Receive buffer 7. Byte 7 of the response data byte. Each response data-byte that is received in the SCIRXSHFT register is transferred to the corresponding register according to the number of bytes received.

### 24.13.26 LIN Mask Register (LINMASK)

Figure 24-54 and Table 24-42 illustrate this register.

**Figure 24-54. LIN Mask Register (LINMASK) [offset = 6Ch]**

31	24	23	16
Reserved		RX ID MASK	
R-0		R/WL-0	
15	8	7	0
Reserved		TX ID MASK	
R-0		R/WL-0	

LEGEND: R/W = Read/Write; R = Read only; WL = Write in LIN mode only; -n = value after reset

**Table 24-42. LIN Mask Register (LINMASK) Field Descriptions**

Bit	Field	Value	Description
31-24	Reserved	0	Reads return 0. Writes have no effect.
23-16	RX ID MASK	0-FFh	Receive ID mask. These bits are effective in LIN mode only. This 8-bit mask is used for filtering an incoming ID message and comparing it to the ID-byte. A compare match of the received ID with the RX ID mask will set the ID RX flag and trigger an ID interrupt if enabled (SET ID INT in SCISSETINT). A 0 bit in the mask indicates that bit is compared to the ID-byte. A 1 bit in the mask indicates that bit is filtered and therefore is not used in the compare.
15-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	TX ID MASK	0-FFh	Transmit ID mask. These bits are effective in LIN mode only. This 8-bit mask is used for filtering an incoming ID message and comparing it to the ID-byte. A compare match of the received ID with the TX ID mask will set the ID TX flag and trigger an ID interrupt if enabled (SET ID INT in SCISSETINT). 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.

**24.13.27 LIN Identification Register (LINID)**

Figure 24-55 and Table 24-43 illustrate this register.

**Figure 24-55. LIN Identification Register (LINID) [offset = 70h]**

31	24	23	16
Reserved		RECEIVED ID	
R-0		R-0	
15	8	7	0
ID-SlaveTask BYTE		ID BYTE	
R/WL-0		R/WL-0	

LEGEND: R/W = Read/Write; R = Read only; WL = Write in LIN mode only; -n = value after reset

**Table 24-43. LIN Identification Register (LINID) Field Descriptions**

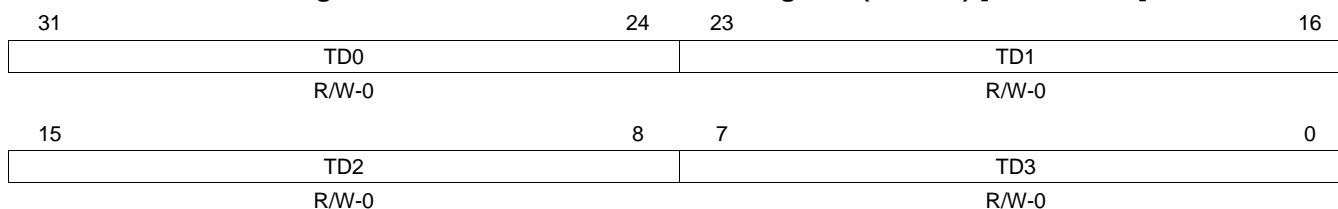
Bit	Field	Value	Description
31-24	Reserved	0	Reads return 0. Writes have no effect.
23-16	RECEIVED ID	0-FFh	Received identification. These bits are effective in LIN mode only. This byte contains the current message identifier. During header reception the received ID is copied from the SCIRXSHF register to this byte if there is no ID-parity error and there has been an RX/TX match.
15-8	ID-SLAVETASK BYTE	0-FFh	ID-SlaveTask Byte. These bits are effective in LIN mode only. This field contains the identifier to which the received ID of an incoming header will be compared to decide whether a receive response, a transmit response, or no action needs to be performed by the LIN node when a header with that particular ID is received.
7-0	ID BYTE	0-FFh	ID byte. This field is effective in LIN mode only. This byte is the LIN mode message ID. On a master node, a write to this register by the CPU initiates a header transmission. For a slave task, this byte is used for message filtering when 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 FFh and the TX ID MASK must be set to FFh.

### 24.13.28 LIN Transmit Buffer 0 Register(LINTD0)

Figure 24-56 and Table 24-44 illustrate this register.

**Figure 24-56. LIN Transmit Buffer 0 Register (LINTD0) [offset = 74h]**



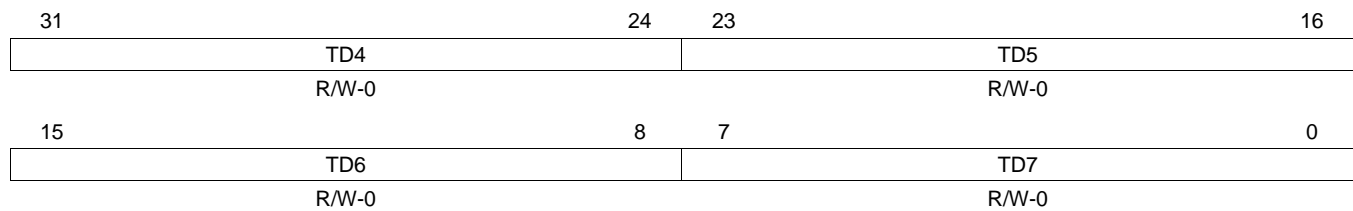
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 24-44. LIN Transmit Buffer 0 Register (LINTD0) Field Descriptions**

Bit	Field	Value	Description
31-24	TD0	0-FFh	8-Bit transmit buffer 0. Byte 0 to be transmitted is written into this register and then copied to SCITXSHF for transmission. Once byte 0 is written in TD0 buffer, transmission will be initiated. <b>Note: TD&lt;x-1&gt; is equivalent to data byte &lt;x&gt; of the LIN frame.</b>
23-16	TD1	0-FFh	8-Bit transmit buffer 1. Byte 1 to be transmitted is written into this register and then copied to SCITXSHF for transmission.
15-8	TD2	0-FFh	8-Bit transmit buffer 2. Byte 2 to be transmitted is written into this register and then copied to SCITXSHF for transmission.
7-0	TD3	0-FFh	8-Bit transmit buffer 3. Byte 3 to be transmitted is written into this register and then copied to SCITXSHF for transmission.

**24.13.29 LIN Transmit Buffer 1 Register (LINTD1)**

Figure 24-57 and Table 24-45 illustrate this register.

**Figure 24-57. LIN Transmit Buffer 1 Register (LINTD1) [offset = 78h]**


LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

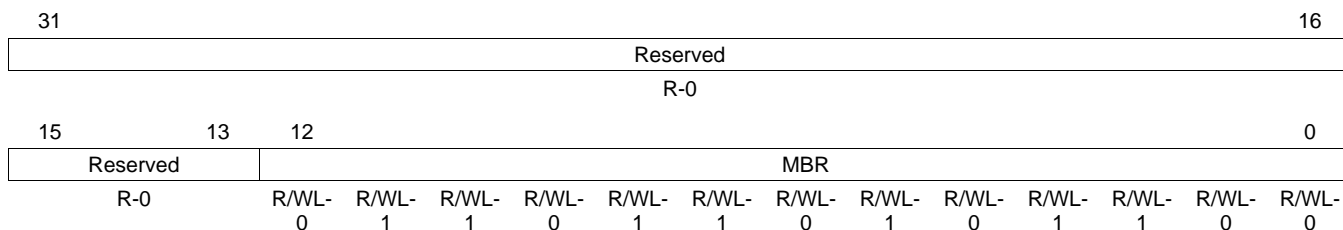
**Table 24-45. LIN Transmit Buffer 1 Register (LINTD1) Field Descriptions**

Bit	Field	Value	Description
31-24	TD4	0-FFh	8-Bit transmit buffer 4. Byte 4 to be transmitted is written into this register and then copied to SCITXSHF for transmission. <b>Note: TD&lt;x-1&gt; is equivalent to data byte &lt;x&gt; of the LIN frame.</b>
23-16	TD5	0-FFh	8-Bit transmit buffer 5. Byte 5 to be transmitted is written into this register and then copied to SCITXSHF for transmission.
15-8	TD6	0-FFh	8-Bit transmit buffer 6. Byte 6 to be transmitted is written into this register and then copied to SCITXSHF for transmission.
7-0	TD7	0-FFh	8-Bit transmit buffer 7. Byte 7 to be transmitted is written into this register and then copied to SCITXSHF for transmission.

### 24.13.30 Maximum Baud Rate Selection Register (MBRS)

Figure 24-58 and Table 24-46 illustrate this register.

**Figure 24-58. Maximum Baud Rate Selection Register (MBRS) [offset = 7Ch]**



LEGEND: R/W = Read/Write; R = Read only; WL = Write in LIN mode only; -n = value after reset

**Table 24-46. Maximum Baud Rate Selection Register (MBRS) Field Descriptions**

Bit	Field	Value	Description
31-13	Reserved	0	Read returns 0. Writes have no effect.
12-0	MBR	0-1FFFh	<p>Maximum baud rate prescaler. This bit is effective in LIN mode only. This 13-bit prescaler is used during the synchronization phase (see <a href="#">Section 24.7.5.2</a>) 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.</p> <p>The MBR value should be programmed to allow a maximum baud rate that is not more than 10% above the expected operating baud rate in the LIN network. Otherwise, a 0x00 data byte could mistakenly be detected as a sync break.</p> <p>The default value for a 70Mhz VCLK is 0xDAC.</p> <p>This MBR prescaler is used by the wake-up and idle time counters for a constant expiration time relative to a 20 kHz rate.</p> <p style="text-align: left; margin-top: 10px;"><math>MBR = \frac{0.9 \times VCLK}{maxbaudrate}</math></p>

(51)

### 24.13.31 Input/Output Error Enable (IODFTCTRL) Register

All the bits in the IODFTCTRL register are used in IODFT (I/O design for test) mode only. [Figure 24-59](#) and [Table 24-47](#) illustrate this register. After the basic SCI/LIN module configuration, enable the required Error mode to be created followed by IODFT Key enable.

**Figure 24-59. Input/Output Error Enable Register (IODFTCTRL) [offset = 90h]**

31	30	29	28	27	26	25	24
BEN	PBEN	CEN	ISFE	Reserved	FEN	PEN	BRKDTENA
R/WL-0	R/WL-0	R/WL-0	R/WL-0	R-0	R/W-0	R/WC-0	R/WC-0
23		21	20	19	18		16
Reserved			PIN SAMPLE MASK			TX SHIFT	
	R-0		R/W-0		R/W-0		
15			12	11			8
Reserved				IODFTENA			
	R-0		R/WP-0	R/WP-1	R/WP-0		R/WP-1
7					2	1	0
Reserved						LPB ENA	RXPENA
	R-0				R/WP-0		R/WP-0

LEGEND: R/W = Read/Write; R = Read only; 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 24-47. Input/Output Error Enable Register (IODFTCTRL) Field Descriptions**

Bit	Field	Value	Description
31	BEN	0	Bit error enable. This bit is effective in LIN mode only. This bit is used to create a bit error. No bit error is created.
		1	The bit received is ORed with 1 and passed to the bit monitor circuitry.
30	PBEN	0	Physical bus error enable. This bit is effective in LIN mode only. This bit is used to create a physical bus error. 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	0	Checksum error enable. This bit is effective in LIN mode only. This bit is used to create a checksum error. No error is created.
		1	The polarity of the CTYPE (checksum type) in the receive checksum calculator is changed so that a checksum error is occurred.
28	ISFE	0	Inconsistent synch field (ISF) error enable. This bit is effective in LIN mode only. This bit is used to create an ISF error. No error is created.
		1	The bit widths in the synch field are varied so that the ISF check fails and the error flag is set.
27	Reserved	0	Read returns 0. Writes have no effect.
26	FEN	0	Frame error enable. This bit is used to create a frame error. No error is created.
		1	The stop bit received is ANDed with 0 and passed to the stop bit check circuitry.



**Table 24-47. Input/Output Error Enable Register (IODFTCTRL) Field Descriptions (continued)**

Bit	Field	Value	Description
25	PEN	0 1	Parity error enable. This bit is effective in SCI-compatible mode only. This bit is used to create a parity error. No parity error occurs. The parity bit received is toggled so that a parity error occurs.
24	BRKD TENA	0 1	Break detect error enable. This bit is effective in SCI-compatible mode only. This bit is used to create a BRKDT error. No error is created. 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 <sub>BITS</sub> so that a BRKDT error occurs.
32-21	Reserved	0	Read returns 0. Writes have no effect.
20-19	PIN SAMPLE MASK	0 1h 2h 3h	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. <b>Note: In IODFT mode testing for pin_sample mask must be done with prescaler P programmed greater than 2 (P &gt; 2).</b> No mask is used. Invert the TX Pin value at TBIT_CENTER. Invert the TX Pin value at TBIT_CENTER + SCLK. Invert the TX Pin value at TBIT_CENTER + 2 SCLK.
18-16	TX SHIFT	0 1h 2h 3h 4h 5h 6h 7h	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. No delay occurs. The value is delayed by 1 SCLK. The value is delayed by 2 SCLK. The value is delayed by 3 SCLK. The value is delayed by 4 SCLK. The value is delayed by 5 SCLK. The value is delayed by 6 SCLK. The value is delayed by 7 SCLK.
15-12	Reserved	0	Read returns 0. Writes have no effect.
11-8	IODFTENA	Ah All Others	IODFT enable key. Write access permitted in Privilege mode only. IODFT is enabled. IODFT is disabled.
7-2	Reserved	0	Read returns 0. Writes have no effect.
1	LPBENA	0 1	Module loopback enable. Write access permitted in Privilege mode only. <b>Note: In analog loopback mode the complete communication path through the I/Os can be tested, whereas in digital loopback mode the I/O buffers are excluded from this path.</b> Digital loopback is enabled. Analog loopback is enabled in module I/O DFT mode when IODFTENA = 1010.
0	RXPENA	0 1	Module analog loopback through receive pin enable. Write access permitted in Privilege mode only. This bit defines whether the I/O buffers for the transmit or the receive pin are included in the communication path (in analog loopback mode). Analog loopback through the transmit pin is enabled. Analog loopback through the receive pin is enabled.

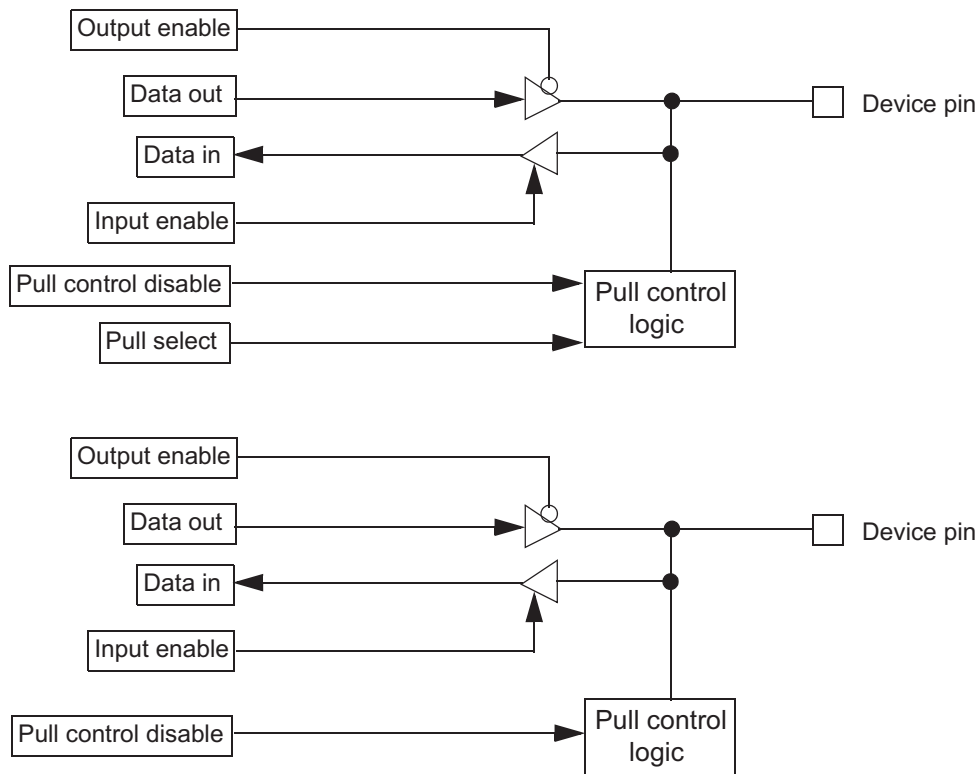
## 24.14 GPIO Functionality

The following sections apply to all device pins that can be configured as functional or general-purpose I/O pins.

### 24.14.1 GPIO Functionality

Figure 24-60 illustrates the GPIO functionality.

Figure 24-60. GPIO Functionality



### 24.14.2 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. The input buffer is enabled.
- Output buffer. The output buffer is disabled.

### 24.14.3 Out of Reset

The following apply if the device is out of reset:

- Pull control. The pull control is enabled by clearing the PD (pull control disable) bit in the SCPIO7 register (Section 24.13.21). In this case, if the PSL (pull select) bit in the SCPIO8 register (Section 24.13.22) is set, the pin will have a pull-up. If the PSL bit is cleared, the pin will have a pull-down. If the PD bit is set in the control register, there is no pull-up or pull-down on the pin.
- Input buffer. The input buffer is permanently enabled in this device.

**NOTE:** The pull-disable logic depends on the pin direction. It is independent of whether the device is in I/O or functional mode. If the pin is configured as output or transmit, then the pulls are disabled automatically.

- Output buffer. A pin can be driven as an output pin if the TX DIR bit is set in the pin direction control register (SCIPIO1; [Section 24.13.15](#)) AND the open-drain feature is not enabled in the SCIPIO6 register ([Section 24.13.20](#)).

#### 24.14.4 Open-Drain Feature Enabled on a Pin

The following apply if the open-drain feature is enabled on a pin.

- The output buffer is enabled if a low signal is being driven on to the pin.
- The output buffer is disabled (the direction control signal DIR is internally forced low) if a high signal is being driven on to the pin.

---

**NOTE:** The open-drain feature is available only in I/O mode (SCIPIO0; [Section 24.13.14](#)).

---

#### 24.14.5 Summary

The behavior of the input buffer, output buffer, and the pull control is summarized in [Table 24-48](#).

**Table 24-48. Input Buffer, Output Buffer, and Pull Control Behavior as GPIO Pins <sup>(1)</sup>**

Device under Reset?	Pin Direction (DIR) <sup>(2)</sup>	Pull Disable (PULDIS) <sup>(3)</sup>	Pull Select (PULSEL) <sup>(4)</sup>	Pull Control	Output Buffer	Input Buffer
Yes	X	X	X	Device- and module-specific	Disabled	Enabled
No	0	0	0	Pull down	Disabled	Enabled
No	0	0	1	Pull up	Disabled	Enabled
No	0	1	0	Disabled	Disabled	Enabled
No	0	1	1	Disabled	Disabled	Enabled
No	1	X	X	Disabled	Enabled	Enabled

<sup>(1)</sup> X = Don't care

<sup>(2)</sup> DIR = 0 for input, 1 for output

<sup>(3)</sup> PULDIS = 0 for enabling pull control= 1 for disabling pull control

<sup>(4)</sup> PULSEL= 0 for pull-down functionality= 1 for pull-up functionality

## ***Serial Communication Interface (SCI) Module***

This chapter contains the description of the serial communication interface (SCI) module.

Topic	Page
<b>25.1 Introduction</b> .....	<b>1450</b>
<b>25.2 SCI Communication Formats</b> .....	<b>1452</b>
<b>25.3 SCI Interrupts</b> .....	<b>1457</b>
<b>25.4 SCI DMA Interface</b> .....	<b>1460</b>
<b>25.5 SCI Configurations</b> .....	<b>1461</b>
<b>25.6 SCI Low Power Mode</b> .....	<b>1463</b>
<b>25.7 SCI Control Registers</b> .....	<b>1465</b>
<b>25.8 GPIO Functionality</b> .....	<b>1501</b>

## 25.1 Introduction

The SCI module is a universal asynchronous receiver-transmitter that implements the standard nonreturn to zero format. The SCI can be used to communicate, for example, through an RS-232 port or over a K-line.

### 25.1.1 SCI Features

The following are the features of the SCI module:

- Standard universal asynchronous receiver-transmitter (UART) communication
- Supports full- or half-duplex operation
- Standard nonreturn to zero (NRZ) format
- Double-buffered receive and transmit functions
- Supports two individually enabled interrupt lines: level 0 and level 1
- Configurable frame format of 3 to 13 bits per character based on the following:
  - Data word length programmable from one to eight bits
  - Additional address bit in address-bit mode
  - Parity programmable for zero or one parity bit, odd or even parity
  - Stop programmable for one or two stop bits
- Asynchronous or isosynchronous communication modes with no CLK pin
- Two multiprocessor communication formats allow communication between more than two devices
- Sleep mode is available to free CPU resources during multiprocessor communication and then wake up to receive an incoming message
- The 24-bit programmable baud rate supports  $2^{24}$  different baud rates provide high accuracy baud rate selection
- Capability to use Direct Memory Access (DMA) for transmit and receive data
- Four error flags and Five status flags provide detailed information regarding SCI events
- Two external pins: SCIRX and SCITX

---

**NOTE:** SCI module does not support UART Hardware Flow Control. This feature can be implemented in Software using a General Purpose I/O pin.

---

### 25.1.2 Block Diagram

Three Major components of the SCI Module are:

- Transmitter
- Baud Clock Generator
- Receiver

**Transmitter (TX)** contains two major registers to perform 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 SCITX pin, one bit at a time.

#### Baud Clock Generator

- A programmable baud generator produces either a baud clock scaled from VBUSP CLK.

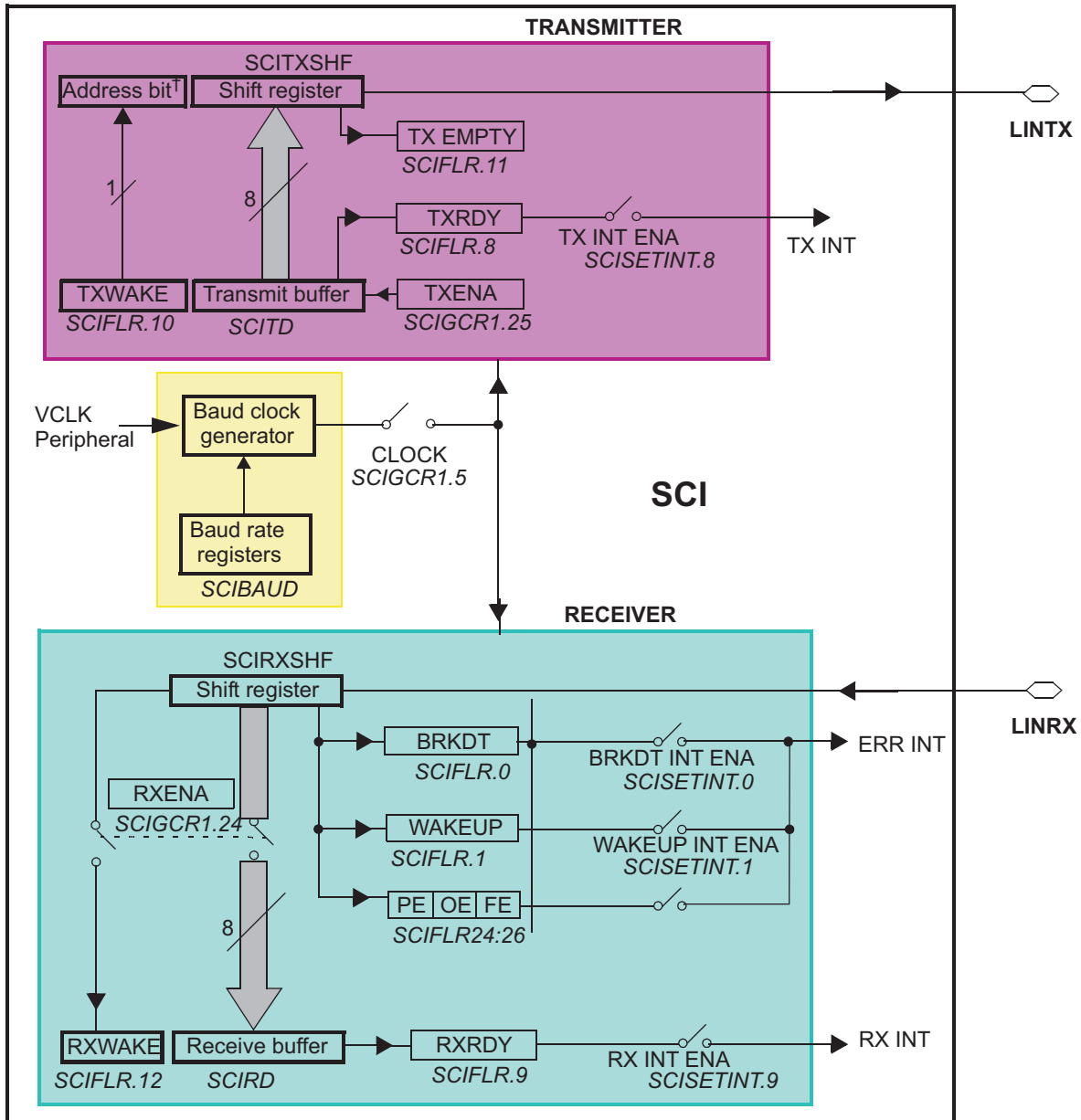
**Receiver (RX)** contains two major registers to perform the double- buffering:

- The receiver shift register (SCIRXSHF) shifts data in from the SCIRX pin one bit at a time and transfers completed data into the receive data buffer.
- The receiver data buffer register (SCIRD) contains received data transferred from the receiver shift register

The SCI receiver and transmitter are double-buffered, and each has its own separate enable and interrupt bits. The receiver and transmitter may each be operated independently or simultaneously in full duplex mode.

To ensure data integrity, the SCI checks the data it receives for breaks, parity, overrun, and framing errors. The bit rate (baud) is programmable to over 16 million different rates through a 24-bit baud-select register. Figure 25-1 shows the detailed SCI block diagram.

Figure 25-1. Detailed SCI Block Diagram



## 25.2 SCI Communication Formats

The SCI module can be configured to meet the requirements of many applications. Because communication formats vary depending on the specific application, many attributes of the SCI are user configurable. The list below describes these configuration options:

- SCI Frame format
- SCI Timing modes
- SCI Baud rate
- SCI Multiprocessor modes

### 25.2.1 SCI Frame Formats

The SCI uses a programmable frame format. All frames consist of the following:

- One start bit
- One to eight data bits
- Zero or one address bit
- Zero or one parity bit
- One or two stop bits

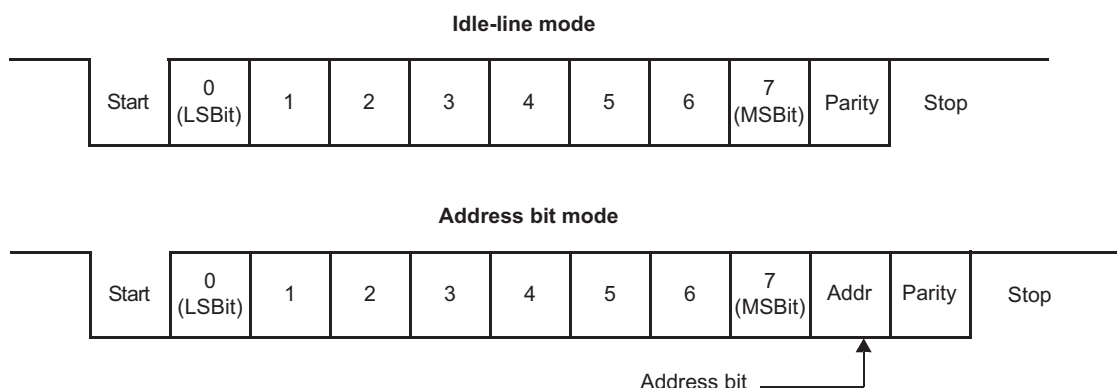
The frame format for both the transmitter and receiver is programmable through the bits in the SCIGCR1 register. Both receive and transmit data is in nonreturn to zero (NRZ) format, which means that the transmit and receive lines are at logic high when idle. Each frame transmission begins with a start bit, in which the transmitter pulls the SCI line low (logic low). Following the start bit, the frame data is sent and received least significant bit first (LSB).

An address bit is present in each frame if the SCI is configured to be in address-bit mode but is not present in any frame if the SCI is configured for idle-line mode. The format of frames with and without the address bit is illustrated in [Figure 25-2](#).

A parity bit is present in every frame when the PARITY ENA bit is set. The value of the parity bit depends on the number of one bits in the frame and whether odd or even parity has been selected via the PARITY ENA bit. Both examples in [Figure 25-2](#) have parity enabled.

All frames include one stop bit, which is always a high level. This high level at the end of each frame is used to indicate the end of a frame to ensure synchronization between communicating devices. Two stop bits are transmitted if the STOP bit in SCIGCR1 register is set. The examples shown in [Figure 25-2](#) use one stop bit per frame.

**Figure 25-2. Typical SCI Data Frame Formats**



### 25.2.2 SCI Timing Mode

The SCI can be configured to use asynchronous or isosynchronous timing using TIMING MODE bit in SCIGCR1 register.

### 25.2.2.1 Asynchronous Timing Mode

The asynchronous timing mode uses only the receive and transmit data lines to interface with devices using the standard universal asynchronous receiver-transmitter (UART) protocol.

In the asynchronous timing mode, each bit in a frame has a duration of 16 SCI baud clock periods. Each bit therefore consists of 16 samples (one for each clock period). When the SCI is using asynchronous mode, the baud rates of all communicating devices must match as closely as possible. Receive errors result from devices communicating at different baud rates.

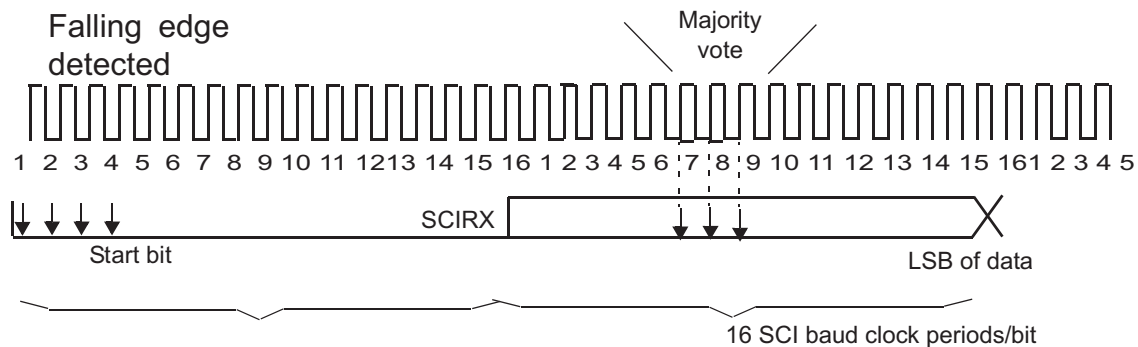
With the receiver in the asynchronous timing mode, the SCI detects a valid start bit if the first four samples after a falling edge on the SCIRX pin are of logic level 0. As soon as a falling edge is detected on SCIRX, the SCI assumes that a frame is being received and synchronizes itself to the bus.

To prevent interpreting noise as Start bit SCI expects SCIRX line to be low for at least four contiguous SCI baud clock periods to detect a valid start bit. The bus is considered idle if this condition is not met. When a valid start bit is detected, the SCI determines the value of each bit by sampling the SCIRX line value during the seventh, eighth, and ninth SCI baud clock periods. A majority vote of these three samples is used to determine the value stored in the SCI receiver shift register. By sampling in the middle of the bit, the SCI reduces errors caused by propagation delays and rise and fall times and data line noises.

Figure 25-3 illustrates how the receiver samples a start bit and a data bit in asynchronous timing mode.

The transmitter transmits each bit for a duration of 16 SCI baud clock periods. During the first clock period for a bit, the transmitter shifts the value of that bit onto the SCITX pin. The transmitter then holds the current bit value on SCITX for 16 SCI baud clock periods.

Figure 25-3. Asynchronous Communication Bit Timing



### 25.2.2.2 Isosynchronous Timing Mode

In isosynchronous timing mode, each bit in a frame has a duration of exactly 1 baud clock period and therefore consists of a single sample. With this timing configuration, the transmitter and receiver are required to make use of the SCICLK pin to synchronize communication with other SCI. **This mode is not fully supported on this device because SCICLK pin is not available.**

### 25.2.3 SCI Baud Rate

The SCI has an internally generated serial clock determined by the peripheral VCLK and the prescalers BAUD. The SCI uses the 24-bit integer prescaler BAUD value of BRSR register to select the required baud rates.

In asynchronous timing mode, the SCI generates a baud clock according to the following formula:

$$\text{Asynchronous baud value} = \frac{\text{VBUSPCLK Frequency}}{16 \text{ BAUD} + 1}$$

For BAUD = 0,

$$\text{Asynchronous baud value} = \frac{\text{VCLK Frequency}}{32} \quad (52)$$



In isosynchronous timing mode, the SCI generates a baud clock according to the following formula:

$$\text{Isosynchronous baud value} = \frac{\text{VBUSPCLK Frequency}}{\text{BAUD} + 1}$$

For BAUD = 0,

$$\text{Isosynchronous baud value} = \frac{\text{VCLK Frequency}}{32} \quad (53)$$

## 25.2.4 SCI Multiprocessor Communication Modes

In some applications, the SCI may be connected to more than one serial communication device. In such a multiprocessor configuration, several frames of data may be sent to all connected devices or to an individual device. In the case of data sent to an individual device, the receiving devices must determine when they are being addressed. When a message is not intended for them, the devices can ignore the following data. When only two devices make up the SCI network, addressing is not needed, so multiprocessor communication schemes are not required.

SCI 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.

---

### NOTE: Avoid Transmitting Simultaneously on the Same Serial Bus

The system designer must ensure that devices connected to the same serial bus line do not attempt to transmit simultaneously. If two devices are transmitting different data, the resulting bus conflict could damage the device..

---

### 25.2.4.1 Idle-Line Multiprocessor Modes

In Idle-line multiprocessor mode, a frame that is preceded by an idle period (10 or more idle bits) is an address frame. A frame that is preceded by fewer than 10 idle bits is a data frame. [Figure 25-4](#) illustrates the format of several blocks and frames with idle-line mode.

There are two ways to transmit an address frame using idle-line mode:

**Method 1:** In software, deliberately leave an idle period between the transmission of the last data frame of the previous block and the address frame of the new block.

**Method 2:** Configure the SCI to automatically send an idle period between the last data frame of the previous block and the address frame of the new block.

Although Method 1 is only accomplished by a delay loop in software, Method 2 can be implemented by using the transmit buffer and the TXWAKE bit in the following manner:

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.

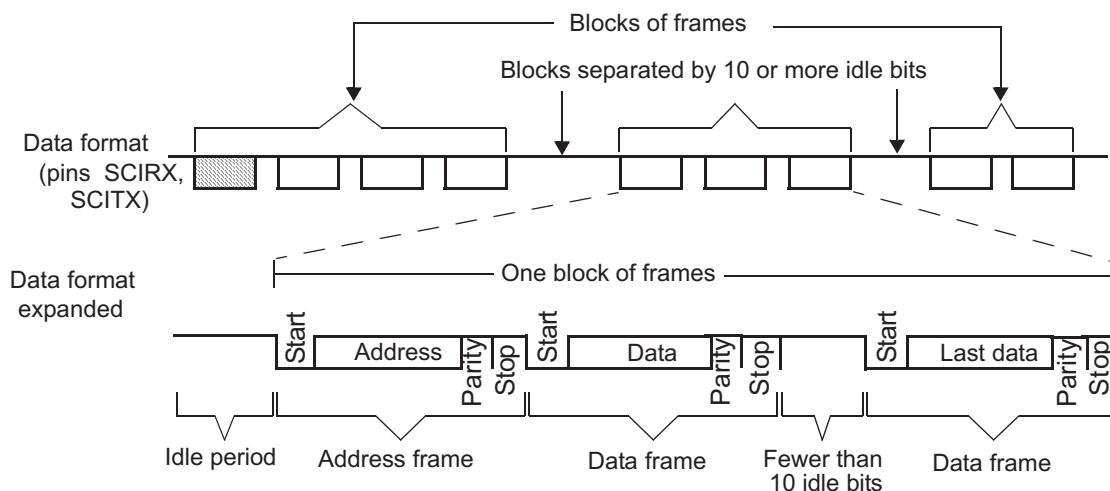
Step3 : Wait for the SCI to clear the TXWAKE flag.

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 25-4. Idle-Line Multiprocessor Communication Format**



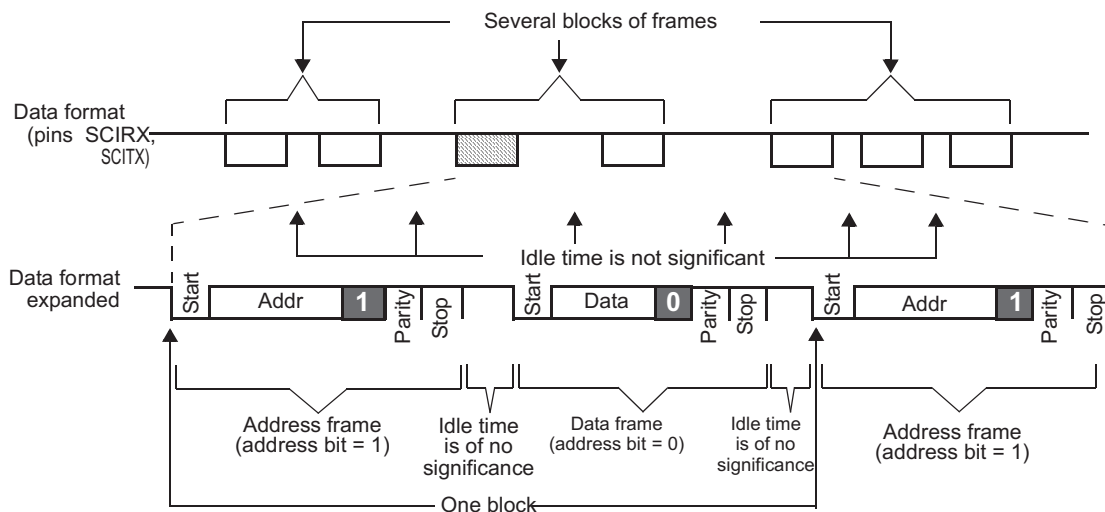
**25.2.4.2 Address-Bit Multiprocessor Mode**

In the address-bit protocol, each frame has an extra bit immediately following the data field called an address bit. A frame with the address bit set to 1 is an address frame; a frame with the address bit set to 0 is a data frame. The idle period timing is irrelevant in this mode. Figure 25-5 illustrates the format of several blocks and frames with the address-bit mode.

When address-bit mode is used, the value of the TXWAKE bit is the value sent as the address bit. To send an address frame, software must set the TXWAKE bit. This bit is cleared as the contents of the SCITD are shifted from the TXWAKE register so that all frames sent are data except when the TXWAKE bit is written as a 1.

No dummy write to SCITD is required before an address frame is sent in address-bit mode. The first byte written to SCITD after the TXWAKE bit is written to 1 is transmitted with the address bit set when address-bit mode is used.

**Figure 25-5. Address-Bit Multiprocessor Communication Format**



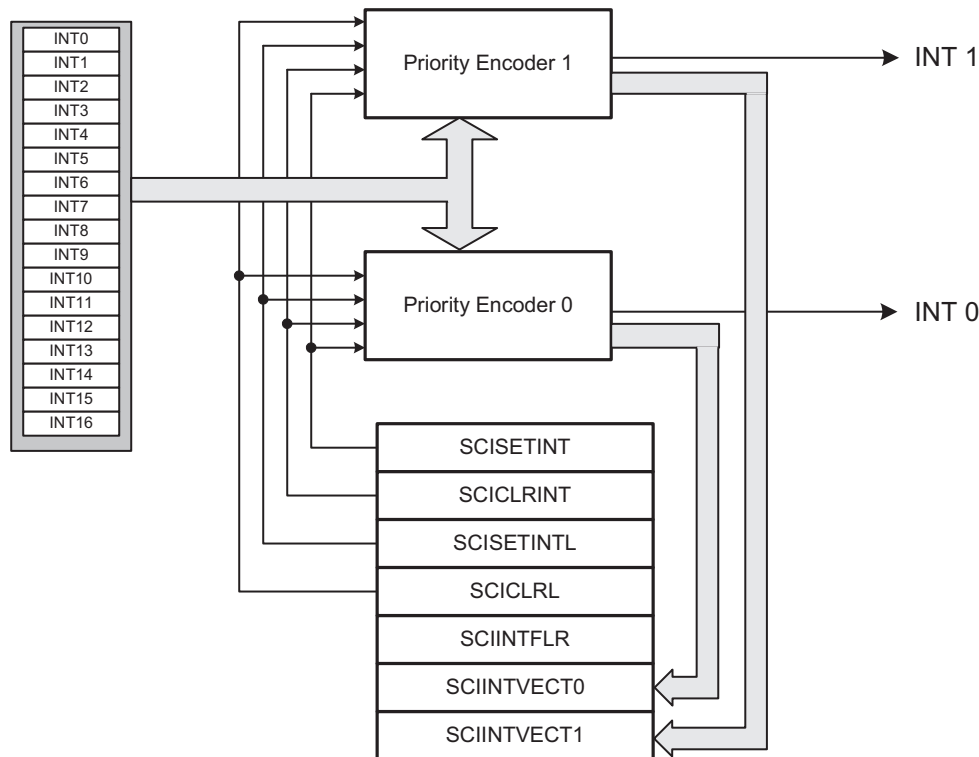
### 25.3 SCI Interrupts

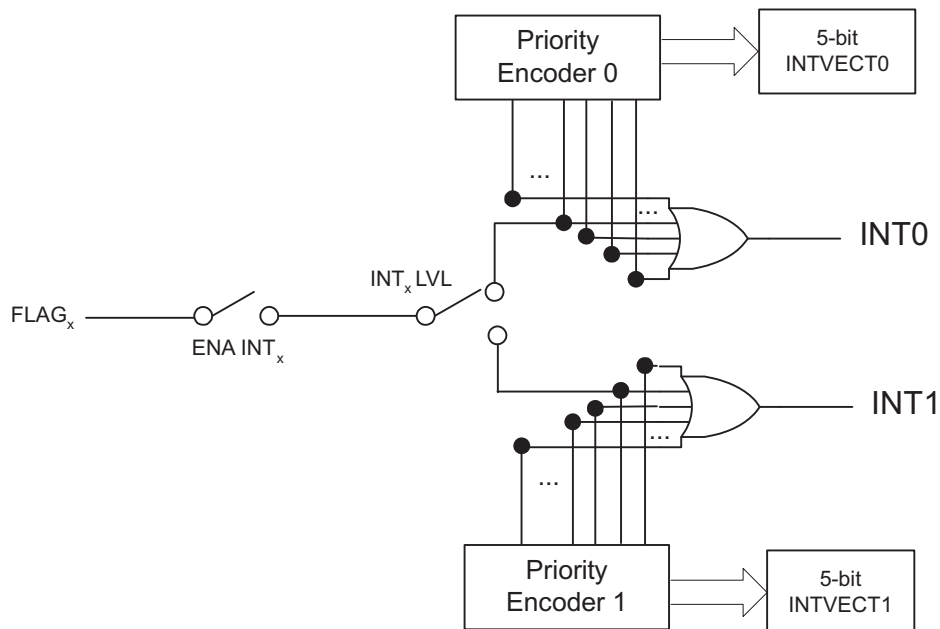
The SCI module has two interrupt lines, level 0 and level 1, to the vectored interrupt manager (VIM) module (see [Figure 25-6](#)). Two offset registers SCIINTVECT0 and SCIINTVECT1 determine which flag triggered the interrupt according to the respective priority encoders. Each interrupt condition has a bit to enable/disable the interrupt in the SCISSETINT 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. SCISSETINTLVL 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 25-6. General Interrupt Scheme**



**Figure 25-7. Interrupt Generation for Given Flags**

### 25.3.1 Transmit Interrupt

To use transmit interrupt functionality, SET TX INT bit must be enabled and SET TX DMA bit must be cleared. The transmit ready (TXRDY) flag is set when the SCI transfers the contents of SCITD to the shift register, SCITXSHF. The TXRDY flag indicates that SCITD is ready to be loaded with more data. In addition, the SCI sets the TX EMPTY bit if both the SCITD and SCITXSHF registers are empty. If the SET TX INT bit is set, then a transmit interrupt is generated when the TXRDY flag goes high. Transmit Interrupt is not generated immediately after setting the SET TX INT bit unlike transmit DMA request. Transmit Interrupt is generated only after the first transfer from SCITD to SCITXSHF, that is first data has to be written to SCITD by the User before any interrupt gets generated. To transmit further data the user can write data to SCITD in the transmit Interrupt service routine.

Writing data to the SCITD register clears the TXRDY bit. When this data has been moved to the SCITXSHF register, the TXRDY bit is set again. The interrupt request can be suspended by setting the CLR TX INT bit; however, when the SET TX INT bit is again set to 1, the TXRDY interrupt is asserted again. The transmit interrupt request can be eliminated until the next series of values is written to SCITD, by disabling the transmitter via the TXENA bit, by a software reset SWnRST, or by a device hardware reset.

### 25.3.2 Receive Interrupt

The receive ready (RXRDY) flag is set when the SCI transfers newly received data from SCIRXSHF to SCIRD. The RXRDY flag therefore indicates that the SCI has new data to be read. Receive interrupts are enabled by the SET RX INT bit. If the SET RX INT is set when the SCI sets the RXRDY flag, then a receive interrupt is generated. The received data can be read in the Interrupt Service routine.

On a device with both SCI and a DMA controller, the bits SET RX DMA ALL and SET RX DMA must be cleared to select interrupt functionality.

### 25.3.3 WakeUp Interrupt

SCI sets the WAKEUP flag if bus activity on the RX line either prevents power-down mode from being entered, or RX line activity causes an exit from power-down mode. If enabled (SET WAKEUP INT), Wakeup interrupt is triggered once WAKEUP flag is set.

### 25.3.4 Error Interrupts

The following error detection are supported with Interrupt by the SCI module:

- Parity errors (PE)
- Frame errors (FE)
- Break Detect errors(BRKDT)
- Overrun errors (OE)

All of these errors (PE,FE, BRKDT,OE) 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. Further details on these flags are explained in SCIFLR register description.

The SCI module supports following 7 interrupts as seen in [Table 25-1](#).

**Table 25-1. SCI Interrupts**

Offset <sup>(1)</sup>	Interrupt
0	Reserved
1	Wakeup
2	Reserved
3	Parity error
4	Reserved
5	Reserved
6	Frame error
7	Break detect error
8	Reserved
9	Overrun error
10	Reserved
11	Receive
12	Transmit
13 - 15	Reserved

<sup>(1)</sup> Offset 1 is the highest priority. Offset 16 is the lowest priority.

## 25.4 SCI DMA Interface

DMA requests for receive (RXDMA request) and transmit (TXDMA request) are available for the SCI module. Refer to the DMA module chapter for DMA module configurations.

### 25.4.1 Receive DMA Requests

This DMA functionality is enabled/disabled by the CPU using the SET RX DMA/CLR RX DMA bits, respectively.

The receiver DMA request is set when a frame is received successfully and DMA functionality has been previously enabled. The RXRDY flag is set when the SCI transfers newly received data from the SCIRXSHF register to the SCIRD buffer. The RXRDY flag therefore indicates that the SCI has new data to be read. Receive DMA requests are enabled by the SET RX INT bit.

Parity, overrun, break detect, wake-up, and framing errors generate an error interrupt request immediately upon detection, if enabled, even if the device is in the process of a DMA data transfer. The DMA transfer is postponed until the error interrupt is served. The error interrupt can delete this particular DMA request by reading the receive buffer.

In multiprocessor mode, the SCI can generate receiver interrupts for address frames and DMA requests for data frames. This is controlled by an extra select bit SET RX DMA ALL.

If the SET RX DMA ALL bit is set and the SET RX DMA bit is set when the SCI sets the RXRDY flag, then a receive DMA request is generated for address and data frames.

If the SET RX DMA ALL bit is cleared and the SET RX DMA bit is set when the SCI sets the RXRDY flag upon receipt of a data frame, then a receive DMA request is generated. Receive interrupt requests are generated for address frames.

In multiprocessor mode with the SLEEP bit set, no DMA is generated for received data frames. The software must clear the SLEEP bit before data frames can be received. [Table 25-2](#) specifies the bit values for DMA requests in multiprocessor modes.

**Table 25-2. DMA and Interrupt Requests in Multiprocessor Modes**

SET RX INT	SET RX DMA	SET RX DMA ALL	ADDR FRAME INT	ADDR FRAME DMA	DATA FRAME INT	DATA FRAME DMA
0	0	x	N	N	N	N
0	1	0	Y	N	N	Y
0	1	1	N	Y	N	Y
1	0	x	Y	N	Y	N
1	1	0	Y	N	Y	Y
1	1	1	Y	Y	Y	Y

In multiprocessor mode, the SCI can generate receiver interrupts for address frames and DMA requests for data frames or DMA requests for both. This is controlled by the SET RX DMA ALL bit.

In multiprocessor mode with the SLEEP bit set, no DMA is generated for received data frames. The software must clear the SLEEP bit before data frames can be received.

### 25.4.2 Transmit DMA Requests

DMA functionality is enabled/disabled by the CPU with SET TX DMA/CLR TX DMA bits, respectively.

The TXRDY flag is set when the SCI transfers the contents of SCITD to SCITXSHF. The TXRDY flag indicates that SCITD is ready to be loaded with more data. In addition, the SCI sets the TX EMPTY bit if both the SCITD and SCITXSHF registers are empty.

Transmit DMA requests are enabled by the setting SET TX DMA and SET TX INT bits.

If the SET TX DMA bit is set, then a TX DMA request is sent to the DMA when data is written to SCITD and TXRDY is set.

The DMA will write the first byte to the transmit buffer.

## 25.5 SCI Configurations

Before the SCI sends or receives data, its registers should be properly configured. Upon power-up or a system-level reset, each bit in the SCI registers is set to a default state. The registers are writable only after the RESET bit 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 SCIRX and SCITX 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 25.5.1](#) / [Section 25.5.2](#))

### 25.5.1 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 SCIRX pin functions as a general purpose I/O pin rather than as an SCI function pin. After a valid idle period is detected, data is automatically received as it arrives on the SCIRX pin.

SCI sets the RXRDY bit when it transfers newly received data from SCIRXSHF to SCIRD. The SCI clears the RXRDY bit after the new data in SCIRD has been read. Also, as data is transferred from SCIRXSHF to SCIRD, the SCI sets 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 unnecessarily 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.

### 25.5.2 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 SCITX pin functions as a general purpose I/O pin rather than as an SCI function pin. Any value written to the SCITD before TXENA is set to 1 is not transmitted. Both of these control bits allow for the SCI transmitter to be held inactive independently of the receiver.



SCI waits for data to be written to SCITD, transfers it to SCITXSHF, and transmits 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.

---

## 25.6 SCI Low Power Mode

The SCI can be put in either local or global low-power mode. Global low-power mode is asserted by the system and is not controlled by the SCI. During global low-power mode, all clocks to the SCI are turned off so the module is completely inactive.

Local low-power mode is asserted by setting the POWERDOWN bit; setting this bit stops the clocks to the SCI internal logic and the module registers. Setting the POWERDOWN bit causes the SCI to enter local low-power mode and clearing the POWERDOWN bit causes SCI to exit from local low-power mode. All the registers are accessible during local power-down mode as any register access enables the clock to SCI for that particular access alone.

The wake-up interrupt is used to allow the SCI to exit low-power mode automatically when a low level is detected on the SCIRX pin and also this clears the POWERDOWN bit. If wake-up interrupt is disabled, then the SCI immediately enters low-power mode whenever it is requested and also any activity on the SCIRX pin does not cause the SCI to exit low-power mode.

---

**NOTE: Enabling Local Low-Power Mode During Receive and Transmit**

If the wake-up interrupt is enabled and low-power mode is requested while the receiver is receiving data, then the SCI immediately generates a wake-up interrupt to clear the 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.

---

### 25.6.1 Sleep Mode for Multiprocessor Communication

When the SCI receives data and transfers that data from SCIRXSHF to SCIRD, the RXRDY bit is set and if RX INT ENA is set, the SCI also generates an interrupt. The interrupt triggers the CPU to read the newly received frame before another one is received. In multiprocessor communication modes, this default behavior may be enhanced to provide selective indication of new data. When SCI receives an address frame that does not match its address, the device can ignore the data following this non-matching address until the next address frame by using sleep mode. Sleep mode can be used with both idle-line and address-bit multiprocessor modes.

If sleep mode is enabled by the SLEEP bit, then the SCI transfers data from SCIRXSHF to SCIRD only for address frames. Therefore, in sleep mode, all data frames are assembled in the SCIRXSHF register without being shifted into the SCIRD and without initiating a receive interrupt or DMA request. Upon reception of an address frame, the contents of the SCIRXSHF are moved into SCIRD, and the software must read SCIRD and determine if the SCI is being addressed by comparing the received address against the address previously set in the software and stored somewhere in memory (the SCI does not have hardware available for address comparison). If the SCI is being addressed, the software must clear the SLEEP bit so that the SCI will load SCIRD with the data of the data frames that follow the address frame.

When the SCI has been addressed and sleep mode has been disabled (in software) to allow the receipt of data, the SCI should check the RXWAKE bit (SCIFLR.12) to determine when the next address has been received. This bit is set to 1 if the current value in SCIRD is an address and set to 0 if SCIRD contains data. If the RXWAKE bit is set, then software should check the address in SCIRD against its own address. If it is still being addressed, then sleep mode should remain disabled. Otherwise, the SLEEP bit should be set again.

Following is a sequence of events typical of sleep mode operation:

- The SCI is configured and both sleep mode and receive actions are enabled.
- An address frame is received and a receive interrupt is generated.
- Software compares the received address frame against that set by software and determines that the SCI is not being addressed, so the value of the SLEEP bit is not changed.
- Several data frames are shifted into SCIRXSHF, but no data is moved to SCIRD and no receive interrupts are generated.
- A new address frame is received and a receive interrupt is generated.
- Software compares the received address frame against that set by software and determines that the

SCI is being addressed and clears the SLEEP bit.

- Data shifted into SCIRXSHF is transferred to SCIRD, and a receive interrupt is generated after each data frame is received.
- In each interrupt routine, software checks RXWAKE to determine if the current frame is an address frame.
- Another address frame is received, RXWAKE is set, software determines that the SCI is not being addressed and sets the SLEEP bit back to 1. No receive interrupts are generated for the data frames following this address frame.

By ignoring data frames that are not intended for the device, fewer interrupts are generated. These interrupts would otherwise require CPU intervention to read data that is of no significance to this specific device. Using sleep mode can help free some CPU resources.

Except for the RXRDY flag, the SCI continues to update the receiver status flags (see [Table 25-6](#)) while sleep mode is active. In this way, if an error occurs on the receive line, an application can immediately respond to the error and take the appropriate corrective action.

Because the RXRDY bit is not updated for data frames when sleep mode is enabled, the SCI can enable sleep mode and use a polling algorithm if desired. In this case, when RXRDY is set, software knows that a new address has been received. If the SCI is not being addressed, then the software should not change the value of the SLEEP bit and should continue to poll RXRDY.

## 25.7 SCI Control Registers

These registers are accessible in 8-, 16-, and 32-bit reads or writes. The SCI is controlled and accessed through the registers listed in [Table 25-3](#). Among the features that can be programmed are the SCI communication and timing modes, baud rate value, frame format, DMA requests, and interrupt configuration.

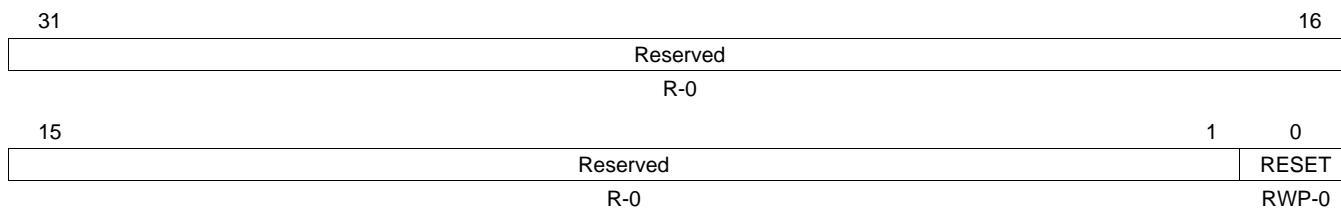
**Table 25-3. SCI Control Registers Summary**

Offset	Acronym	Register Description	Section
00	SCIGCR0	SCI Global Control Register 0	<a href="#">Section 25.7.1</a>
04h	SCIGCR1	SCI Global Control Register 1	<a href="#">Section 25.7.2</a>
0Ch	SCISSETINT	SCI Set Interrupt Register	<a href="#">Section 25.7.3</a>
10h	SCICLEARINT	SCI Clear Interrupt Register	<a href="#">Section 25.7.4</a>
14h	SCISSETINTLVL	SCI Set Interrupt Level Register	<a href="#">Section 25.7.5</a>
18h	SCICLEARINTLVL	SCI Clear Interrupt Level Register	<a href="#">Section 25.7.6</a>
1Ch	SCIFLR	SCI Flags Register	<a href="#">Section 25.7.7</a>
20h	SCIINTVECT0	SCI Interrupt Vector Offset 0	<a href="#">Section 25.7.8</a>
24h	SCIINTVECT1	SCI Interrupt Vector Offset 1	<a href="#">Section 25.7.9</a>
28h	SCIFORMAT	SCI Format Control Register	<a href="#">Section 25.7.10</a>
2Ch	BRS	Baud Rate Selection Register	<a href="#">Section 25.7.11</a>
30h	SCIED	Receiver Emulation Data Buffer	<a href="#">Section 25.7.12.1</a>
34h	SCIRD	Receiver Data Buffer	<a href="#">Section 25.7.12.2</a>
38h	SCITD	Transmit Data Buffer	<a href="#">Section 25.7.12.3</a>
3Ch	SCIPIO0	SCI Pin I/O Control Register 0	<a href="#">Section 25.7.13</a>
40h	SCIPIO1	SCI Pin I/O Control Register 1	<a href="#">Section 25.7.14</a>
44h	SCIPIO2	SCI Pin I/O Control Register 2	<a href="#">Section 25.7.15</a>
48h	SCIPIO3	SCI Pin I/O Control Register 3	<a href="#">Section 25.7.16</a>
4Ch	SCIPIO4	SCI Pin I/O Control Register 4	<a href="#">Section 25.7.17</a>
50h	SCIPIO5	SCI Pin I/O Control Register 5	<a href="#">Section 25.7.18</a>
54h	SCIPIO6	SCI Pin I/O Control Register 6	<a href="#">Section 25.7.19</a>
58h	SCIPIO7	SCI Pin I/O Control Register 7	<a href="#">Section 25.7.20</a>
5Ch	SCIPIO8	SCI Pin I/O Control Register 8	<a href="#">Section 25.7.21</a>
90h	IODFTCTRL	Input/Output Error Enable Register	<a href="#">Section 25.7.22</a>

### 25.7.1 SCI Global Control Register 0 (SCIGCR0)

The SCIGCR0 register defines the module reset. [Figure 25-8](#) and [Table 25-4](#) illustrate this register.

**Figure 25-8. SCI Global Control Register 0 (SCIGCR0) [offset = 00]**



LEGEND: R/W = Read/Write; R = Read only; RWP = Read/Write in privileged mode only; -n = value after reset

**Table 25-4. SCI Global Control Register 0 (SCIGCR0) Field Descriptions**

Bit	Field	Value	Description
31-1	Reserved	0	Read returns 0. Writes have no effect.
0	RESET	0	This bit resets the SCI module. SCI module is in reset.
		1	SCI module is out of reset.
			<b>Note: Read/Write in privileged mode only.</b>

### 25.7.2 SCI Global Control Register 1 (SCIGCR1)

The SCIGCR1 register defines the frame format, protocol, and communication mode used by the SCI. Figure 25-9 and Table 25-5 illustrate this register.

**Figure 25-9. SCI Global Control Register 1 (SCIGCR1) [offset = 04h]**

31				26				25		24					
Reserved						TXENA		RXENA							
R-0						R/W-0		R/W-0							
23				18				17		16					
Reserved						CONT		LOOP BACK							
R-0						R/W-0		R/W-0							
15				10				9		8					
Reserved						POWERDOWN		SLEEP							
R-0						R/WL-0		R/W-0							
7		6		5		4		3		2		1		0	
SW nRST		Reserved		CLOCK		STOP		PARITY		PARITY ENA		TIMING MODE		COMM MODE	
R/W-0		R-0		R/W-0		R/WC-0		R/WC-0		R/W-0		R/WC-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; WC = Write in sci-compatible mode only; -n = value after reset

**NOTE:** The SCIGCR1 Control Register Bits should not be changed during Frame Transmission or Reception..

**Table 25-5. SCI Global Control Register 1 (SCIGCR1) Field Descriptions**

Bit	Field	Value	Description
31-26	Reserved	0	Read returns 0. Writes have no effect.
25	TXENA	0	Transmit enable. Data is transferred from SCITD to the SCITXSHF shift out register only when the TXENA bit is set.
		1	Disable transfers from SCITD to SCITXSHF. Enable SCI to transfer data from SCITD to SCITXSHF. <b>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.</b>
24	RXENA	0	Receive enable. RXENA allows or prevents the transfer of data from SCIRXSHF to SCIRD.
		1	The receiver will not transfer data from the shift buffer to the receive buffer. The receiver will transfer data from the shift buffer to the receive buffer. <b>Note: Clearing RXENA stops received characters from being transferred into the receive buffer or multi-buffers, prevents the RX status flags (see Table 25-4) 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.</b> <b>Note: If RXENA is cleared before a frame is completely received, the data from the frame is not transferred into the receive buffer.</b> <b>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.</b>
23-18	Reserved	0	Read returns 0. Writes have no effect.
17	CONT	0	Continue on suspend. This bit has an effect only when a program is being debugged with an emulator, and it determines how the SCIOperates when the program is suspended. The
		1	When debug mode is entered, the SCI state machine is frozen. Transmissions are halted and resume when debug mode is exited. When debug mode is entered, the SCI continues to operate until the current transmit and receive functions are complete.

**Table 25-5. SCI Global Control Register 1 (SCIGCR1) Field Descriptions (continued)**

Bit	Field	Value	Description
16	LOOP BACK	0 1	<p>Loopback bit. The self-checking option for the SCI can be selected with this bit. If the SCITX and SCIRX pins are configured with SCI functionality, then the SCITX pin is internally connected to the SCIRX pin. Externally, during loop back operation, the SCITX pin outputs a high value and the SCIRX pin is in a high-impedance state. If this bit value is changed while the SCI is transmitting or receiving data, errors may result.</p> <p>0 Loop back mode is disabled. 1 Loop back mode is enabled.</p>
15-10	Reserved	0	Read returns 0. Writes have no effect.
9	POWERDOWN	0 1	<p>Power down. When the POWERDOWN bit is set, the SCI attempts to enter local low-power mode. If the POWERDOWN bit is set while the receiver is actively receiving data and the wake-up interrupt is enabled, then the SCI immediately asserts an error interrupt to prevent low-power mode from being entered. Only Privilege mode writes allowed.</p> <p>0 Automatic baud rate adjustment is disabled. 1 Automatic baud rate adjustment is enabled.</p>
8	SLEEP	0 1	<p>SCI sleep. In a multiprocessor configuration, this bit controls the receive sleep function. Clearing this bit brings the SCI out of sleep mode.</p> <p>0 Sleep mode is disabled. 1 Sleep mode is enabled.</p> <p><b>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 25-4) 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.</b></p> <p><b>Note: The SLEEP bit is not automatically cleared when an address byte is detected.</b></p> <p>See Section 25.6 for more information on using the SLEEP bit for multiprocessor communication.</p>
7	SWnRST	0 1	<p>Software reset (active low). This bit is effective in LIN and SCI modes.</p> <p>0 The SCI is in its reset state; no data will be transmitted or received. Writing a 0 to this bit initializes the SCI state machines and operating flags as defined in Table 25-6 and Table 25-7. All affected logic is held in the reset state until a 1 is written to this bit.</p> <p>1 The SCI is in its ready state; transmission and reception can be done. After this bit is set to 1, the configuration of the module should not change.</p> <p><b>Note: The SCI should only be configured while SWnRESET = 0.</b></p>
6	Reserved	0	Read returns 0. Writes have no effect.
5	CLOCK	0 1	<p>SCI internal clock enable. The CLOCK bit determines the source of the module clock on the SCICLK pin.</p> <p>0 The external SCICLK is the clock source. 1 The internal SCICLK is the clock source.</p> <p><b>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.</b></p>
4	STOP	0 1	<p>SCI number of stop bits per frame.</p> <p>0 One stop bit is used. 1 Two stop bits are used.</p> <p><b>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.</b></p>

**Table 25-5. SCI Global Control Register 1 (SCIGCR1) Field Descriptions (continued)**

Bit	Field	Value	Description
3	PARITY	0	SCI parity odd/even selection. If the PARITY ENA bit is set, PARITY designates odd or even parity. Odd parity is used.
		1	Even parity is used.  <b>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.</b>  <b>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.</b>  <b>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.</b>
2	PARITY ENA	0	Parity enable. This bit enables or disables the parity function. Parity is disabled; no parity bit is generated during transmission or is expected during reception.
		1	Parity is enabled. A parity bit is generated during transmission and is expected during reception.
1	TIMING MODE	0	SCI timing mode bit. Synchronous timing is used.
		1	Asynchronous timing is used.
0	COMM MODE	1	SCI communication mode bit. Idle-line mode is used.
		0	Address-bit mode is used.

**Table 25-6. SCI Receiver Status Flags**

SCI Flag	Register	Bit	Value After SW nRESET <sup>(1)</sup>
FE	SCIFLR	26	0
OE	SCIFLR	25	0
PE	SCIFLR	24	0
RXWAKE	SCIFLR	12	0
RXRDY	SCIFLR	9	0
BRKDT	SCIFLR	0	0

<sup>(1)</sup> The flags are frozen with their reset value while SW nRESET = 0.

**Table 25-7. SCI Transmitter Status Flags**

SCI Flag	Register	Bit	Value After SW nRESET <sup>(1)</sup>
TX EMPTY	SCIFLR	11	1
TXRDY	SCIFLR	8	1

<sup>(1)</sup> The flags are frozen with their reset value while SW nRESET = 0.



### 25.7.3 SCI Set Interrupt Register (SCISSETINT)

Figure 25-10 and Table 25-8 illustrate this register. SCISSETINT register is used to enable the required interrupts supported by the module.

**Figure 25-10. SCI Set Interrupt Register (SCISSETINT) [offset = 0Ch]**

31	27	26	25	24
Reserved		SET FE INT	SET OE INT	SET PE INT
R-0		R/W-0	R/W-0	R/W-0
23	19	18	17	16
Reserved		SET RX DMA ALL	SET RX DMA	SET TX DMA
R-0		R/WC-0	R/W-0	R/W-0
15		10	9	8
Reserved			SET RX INT	SET TX INT
R-0			R/W-0	R/W-0
7		2	1	0
Reserved			SET WAKE UP INT	SET BRKDT INT
R-0			R/W-0	R/WC-0

LEGEND: R/W = Read/Write; R = Read only; WC = Write in sci-compatible mode only; -n = value after reset

**Table 25-8. SCI Set Interrupt Register (SCISSETINT) Field Descriptions**

Bit	Field	Value	Description
31-27	Reserved	0	Read returns 0. Writes have no effect.
26	SET FE INT	0	Set framing-error interrupt. Setting this bit enables the SCI module to generate an interrupt when a framing error occurs. <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read or write:</i> The interrupt is enabled.
25	SET OE INT	0	Set overrun-error interrupt. Setting this bit enables the SCI module to generate an interrupt when an overrun error occurs. <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read or write:</i> The interrupt is enabled.
24	SET PE INT	0	Set parity interrupt. Setting this bit enables the SCI module to generate an interrupt when a parity error occurs. <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read or write:</i> The interrupt is enabled.
23-19	Reserved	0	Read returns 0. Writes have no effect.
18	SET RX DMA ALL	0	Set receive DMA all. This bit determines if a separate interrupt is generated for the address frames sent in multiprocessor communications. When this bit is 0, RX interrupt requests are generated for address frames and DMA requests are generated for data frames. When this bit is 1, RX DMA requests are generated for both address and data frames. <i>Read:</i> The DMA request is disabled for address frames (the receive interrupt request is enabled for address frames). <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read and write:</i> The DMA request is enabled for address and data frames
17	SET RX DMA	0	Set receiver DMA. To enable receiver DMA requests, this bit must be set. If it is cleared, interrupt requests are generated depending on bit SCISSETINT. <i>Read:</i> The DMA request is disabled. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read and write:</i> The DMA request is enabled for address and data frames

**Table 25-8. SCI Set Interrupt Register (SCISSETINT) Field Descriptions (continued)**

Bit	Field	Value	Description
16	SET TX DMA	0	Set transmit DMA. To enable DMA requests for the transmitter, this bit must be set. If it is cleared, interrupt requests are generated depending on SET TX INT bit (SCISSETINT). <i>Read:</i> Transmit DMA request is disabled. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read and write:</i> Transmit DMA request is enabled.
15-10	Reserved	0	Read returns 0. Writes have no effect.
9	SET RX INT	0	Receiver interrupt enable. Setting this bit enables the SCI to generate a receive interrupt after a frame has been completely received and the data is being transferred from SCIRXSHF to SCIRD. <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read or write:</i> The interrupt is enabled.
8	SET TX INT	0	Set transmitter interrupt. Setting this bit enables the SCI to generate a transmit interrupt as data is being transferred from SCITD to SCITXSHF and the TXRDY bit is being set. <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read or write:</i> The interrupt is enabled.
7-2	Reserved	0	Read returns 0. Writes have no effect.
1	SET WAKEUP INT	0	Set wakeup interrupt. Setting this bit enables the SCI to generate a wakeup interrupt and thereby exit lowpower mode. If enabled, the wakeup interrupt is asserted when local lowpower mode is requested while the receiver is busy or if a low level is detected on the SCIRX pin during lowpower mode. <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read or write:</i> The interrupt is enabled.
0	SET BRKDT INT	0	Set breakdetect interrupt. Setting this bit enables the SCI to generate an error interrupt if a break condition is detected on the SCIRX pin. <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read or write:</i> The interrupt is enabled.

### 25.7.4 SCI Clear Interrupt Register (SCICLEARINT)

Figure 25-11 and Table 25-9 illustrate this register. : SCICLEARINT register is used to clear the selected enabled interrupts with out accessing SCISSETINT register.

**Figure 25-11. SCI Clear Interrupt Register (SCICLEARINT) [offset = 10h]**

31	27	26	25	24
Reserved		CLR FE INT	CLR OE INT	CLR PE INT
R-0		R/W-0	R/W-0	R/W-0
23	19	18	17	16
Reserved		CLR RX DMA ALL	CLR RX DMA	CLR TX DMA
R-0		R/WC-0	R/W-0	R/W-0
15		10	9	8
Reserved			CLR RX INT	CLR TX INT
R-0			R/W-0	R/W-0
7		2	1	0
Reserved			CLR WAKE UP INT	CLR BRKDT INT
R-0			R/W-0	R/WC-0

LEGEND: R/W = Read/Write; R = Read only; WC = Write in sci-compatible mode only; -n = value after reset

**Table 25-9. SCI Clear Interrupt Register (SCICLEARINT) Field Descriptions**

Bit	Field	Value	Description
31-27	Reserved	0	Read returns 0. Writes have no effect.
26	CLR FE INT	0	Clear framing-error interrupt. This bit disables the framing-error interrupt when set. <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The interrupt is enabled. <i>Write:</i> The interrupt is disabled.
25	CLR CE INT	0	Clear overrun-error interrupt. This bit disables the SCI overrun error interrupt when set. <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The interrupt is enabled. <i>Write:</i> The interrupt is disabled.
24	CLR PE INT	0	Clear parity interrupt. This bit disables the parity error interrupt when set. <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The interrupt is enabled. <i>Write:</i> The interrupt is disabled.
23-19	Reserved	0	Read returns 0. Writes have no effect.
18	CLR RX DMA ALL	0	Clear receive DMA all. This bit clears the receive DMA request for address frames when set. Only receive data frames generate a DMA request. <i>Read:</i> 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. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The receive DMA request for address and data frames is enabled. <i>Write:</i> The receive DMA request for address and data frames is disabled.

**Table 25-9. SCI Clear Interrupt Register (SCICLEARINT) Field Descriptions (continued)**

Bit	Field	Value	Description
17	CLR RX DMA	0	Clear receive DMA request. This bit disables the receive DMA request when set. <i>Read:</i> The DMA request is disabled. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The receive DMA request is enabled. <i>Write:</i> The receive DMA request for is disabled.
16	CLR TX DMA	0	Clear transmit DMA request. This bit disables the transmit DMA request when set. <i>Read:</i> Transmit DMA request is disabled. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The transmit DMA request is enabled. <i>Write:</i> The transmit DMA request for is disabled.
15-10	Reserved	0	Read returns 0. Writes have no effect.
9	CLR RX INT	0	Clear receiver interrupt. This bit disables the receiver interrupt when set. <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The interrupt is enabled. <i>Write:</i> The interrupt is disabled.
8	CLR TX INT	0	Clear transmitter interrupt. This bit disables the transmitter interrupt when set. <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The interrupt is enabled. <i>Write:</i> The interrupt is disabled.
7-2	Reserved	0	Read returns 0. Writes have no effect.
1	CLR WAKEUP INT	0	Clear wakeup interrupt. This bit disables the wakeup interrupt when set. <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The interrupt is enabled. <i>Write:</i> The interrupt is disabled.
0	CLR BRKDT INT	0	Clear breakdetect interrupt. This bit disables the break-detect interrupt when set. <i>Read:</i> The interrupt is disabled. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The interrupt is enabled. <i>Write:</i> The interrupt is disabled.

### 25.7.5 SCI Set Interrupt Level Register (SCISSETINTLVL)

Figure 25-12 and Table 25-10 illustrate this register. This register is used to set the interrupt level for the supported interrupts.

**Figure 25-12. SCI Set Interrupt Level Register (SCISSETINTLVL) [offset = 14h]**

31	27	26	25	24
Reserved		SET FE INT LVL	SET OE INT LVL	SET PE INT LVL
R-0		R/W-0	R/W-0	R/W-0
23	19	18	17	16
Reserved		SET RX DMA ALL INT LVL	Reserved	
R-0		R/WC-0	R-0	
15			10	9
Reserved			SET RX INT LVL	SET TX INT LVL
R-0			R/W-0	R/W-0
7			2	1
Reserved			SET WAKE UP INT LVL	SET BRKDT INT LVL
R-0			R/W-0	R/WC-0

LEGEND: R/W = Read/Write; R = Read only; WC = Write in sci-compatible mode only; -n = value after reset

**Table 25-10. SCI Set Interrupt Level Register (SCISSETINTLVL) Field Descriptions**

Bit	Field	Value	Description
31-27	Reserved	0	Read returns 0. Writes have no effect.
26	SET FE INT LVL	0	Set framing-error interrupt level. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read or write:</i> The interrupt level is mapped to the INT1 line.
25	SET CE INT LVL	0	Set overrun-error interrupt level. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read or write:</i> The interrupt level is mapped to the INT1 line.
24	SET PE INT LVL	0	Set parity error interrupt level. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read or write:</i> The interrupt level is mapped to the INT1 line.
23-19	Reserved	0	Read returns 0. Writes have no effect.
18	SET RX DMA ALL LVL	0	Set receive DMA all interrupt levels. <i>Read:</i> The receive interrupt request for address frames is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read or write:</i> The receive interrupt request for address frames is mapped to the INT1 line.
17-14	Reserved	0	Read returns 0. Writes have no effect.
9	SET RX INT LVL	0	Set receiver interrupt level. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read or write:</i> The interrupt level is mapped to the INT1 line.

**Table 25-10. SCI Set Interrupt Level Register (SCISSETINTLVL) Field Descriptions (continued)**

Bit	Field	Value	Description
8	SET TX INT LVL	0	Set transmitter interrupt level. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read or write:</i> The interrupt level is mapped to the INT1 line.
7-2	Reserved	0	Read returns 0. Writes have no effect.
1	SET WAKEUP INT LVL	0	Set wakeup interrupt level. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read or write:</i> The interrupt level is mapped to the INT1 line.
0	SET BRKDT INT LVL	0	Set breakdetect interrupt level. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read or write:</i> The interrupt level is mapped to the INT1 line.

### 25.7.6 SCI Clear Interrupt Level Register (SCICLEARINTLVL)

Figure 25-13 and Table 25-11 illustrate this register.

**Figure 25-13. SCI Clear Interrupt Level Register (SCICLEARINTLVL) [offset = 18h]**

31	27	26	25	24
Reserved		CLR FE INT LVL	CLR OE INT LVL	CLR PE INT LVL
R-0		R/W-0	R/W-0	R/W-0
23	19	18	17	16
Reserved		CLR RX DMA ALL INT LVL	Reserved	
R-0		R/WC-0	R-0	
15			10	9
Reserved			CLR ID RX INT LVL	CLR TX INT LVL
R-0			R/W-0	R/W-0
7			2	1
Reserved			CLR WAKE UP INT LVL	CLR BRKDT INT LVL
R-0			R/W-0	R/WC-0

LEGEND: R/W = Read/Write; R = Read only; WC = Write in sci-compatible mode only; -n = value after reset

**Table 25-11. SCI Clear Interrupt Level Register (SCICLEARINTLVL) Field Descriptions**

Bit	Field	Value	Description
31-27	Reserved	0	Read returns 0. Writes have no effect.
26	CLR FE INT LVL	0	Clear framing-error interrupt. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The interrupt level is mapped to the INT1 line. <i>Write:</i> The interrupt level is mapped to the INT0 line.
25	CLR CE INT LVL	0	Clear overrun-error interrupt. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The interrupt level is mapped to the INT1 line. <i>Write:</i> The interrupt level is mapped to the INT0 line.
24	CLR PE INT LVL	0	Clear parity interrupt. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The interrupt level is mapped to the INT1 line. <i>Write:</i> The interrupt level is mapped to the INT0 line.
23-19	Reserved	0	Read returns 0. Writes have no effect.
18	CLR RX DMA ALL LVL	0	Clear receive DMA interrupt level. <i>Read:</i> The receive interrupt request for address frames is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The receive interrupt request for address frames is mapped to the INT1 line. <i>Write:</i> The receive interrupt request for address frames is mapped to the INT0 line.
17-10	Reserved	0	Read returns 0. Writes have no effect.

**Table 25-11. SCI Clear Interrupt Level Register (SCICLEARINTLVL) Field Descriptions (continued)**

Bit	Field	Value	Description
9	CLR RX INT LVL	0	Clear receiver interrupt. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The interrupt level is mapped to the INT1 line. <i>Write:</i> The interrupt level is mapped to the INT0 line.
8	CLR TX INT LVL	0	Clear transmitter interrupt. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The interrupt level is mapped to the INT1 line. <i>Write:</i> The interrupt level is mapped to the INT0 line.
7-2	Reserved	0	Read returns 0. Writes have no effect.
1	CLR WAKEUP INT LVL	0	Clear wakeup interrupt. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The interrupt level is mapped to the INT1 line. <i>Write:</i> The interrupt level is mapped to the INT0 line.
0	CLR BRKDT INT LVL	0	Clear breakdetect interrupt. <i>Read:</i> The interrupt level is mapped to the INT0 line. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The interrupt level is mapped to the INT1 line. <i>Write:</i> The interrupt level is mapped to the INT0 line.



### 25.7.7 SCI Flags Register (SCIFLR)

Figure 25-14 and Table 25-12 illustrate this register.

**Figure 25-14. SCI Flags Register (SCIFLR) [offset = 1Ch]**

31	Reserved			26	FE	25	OE	24	PE
	R-0				R/W-0		R/W-0		R/W-0
23	Reserved							16	
	R-0								
15	Reserved		13	12	11	10	9	8	
	R-0			R/WC-0	R/W-1	R/WC-0	R/W-0	R/W-1	
7	Reserved			4	3	2	1	0	
	R-0				R/W-0	R-0	R/WL-0	R/WC-0	

LEGEND: R/W = Read/Write; R = Read only; WC = Write in sci-compatible mode only; -n = value after reset

**Table 25-12. SCI Flags Register (SCIFLR) Field Descriptions**

Bit	Field	Value	Description
31-27	Reserved		Read returns 0. Writes have no effect.
26	FE	0 1	<p>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 SCISSETINT (SCI Set Interrupt Register (SCISSETINT)). The framing error flag is cleared by the following:</p> <ul style="list-style-type: none"> <li>Setting of the SW nRST bit</li> <li>Setting of the RESET bit</li> <li>A system reset</li> <li>Writing a 1 to this bit</li> <li>Reading the corresponding interrupt offset in SCIINTVECT0/1</li> <li>Reception of a new character/frame, depending on whether the module is in SCI compatible or LIN mode</li> </ul> <p>In multibuffer mode the frame is defined in the SCIFORMAT register.</p> <p><b>0</b> <i>Read:</i> No framing error has been detected since the last clear. <i>Write:</i> Writing a 0 to this bit has no effect.</p> <p><b>1</b> <i>Read:</i> A framing error has been detected since the last clear. <i>Write:</i> The bit is cleared to 0.</p>
25	OE	0 1	<p>Overrun error flag. This bit is set when the transfer of data from SCIRXSHF to SCIRD overwrites unread data already in SCIRD. Detection of an overrun error causes the LIN to generate an error interrupt if the SET OE INT bit = 1. The OE flag is reset by the following:</p> <ul style="list-style-type: none"> <li>Setting of the SW nRST bit</li> <li>Setting of the RESET bit</li> <li>A system reset</li> <li>Writing a 1 to this bit</li> <li>Reading the corresponding interrupt offset in SCIINTVECT0/1</li> </ul> <p><b>0</b> <i>Read:</i> No overrun error has been detected since the last clear. <i>Write:</i> Writing a 0 to this bit has no effect.</p> <p><b>1</b> <i>Read:</i> An overrun error has been detected since the last clear. <i>Write:</i> The bit is cleared to 0.</p>

**Table 25-12. SCI Flags Register (SCIFLR) Field Descriptions (continued)**

Bit	Field	Value	Description
24	PE	<p>0</p> <p>1</p>	<p>Parity error flag. This bit is set when a parity error is detected in the received data. In SCI address-bit mode, the parity is calculated on the data and address bit fields of the received frame. In idle-line mode, only the data is used to calculate parity. An error is generated when a character is received with a mismatch between the number of 1s and its parity bit. If the parity function is disabled (SCIGCR[2] = 0), the PE flag is disabled and read as 0. Detection of a parity error causes the LIN to generate an error interrupt if the SET PE INT bit = 1. The PE bit is reset by the following:</p> <ul style="list-style-type: none"> <li>• Setting of the SW nRST bit</li> <li>• Setting of the RESET bit</li> <li>• A system reset</li> <li>• Writing a 1 to this bit</li> <li>• Reception of a new character or frame, depending on whether the module is in SCI compatible or LIN mode, respectively.</li> <li>• Reading the corresponding interrupt offset in SCIINTVECT0/1</li> </ul> <p><i>Read:</i> No parity error has been detected since the last clear. <i>Write:</i> Writing a 0 to this bit has no effect.</p> <p><i>Read:</i> A parity error has been detected since the last clear. <i>Write:</i> The bit is cleared to 0.</p>
23-13	Reserved	0	Read returns 0. Writes have no effect.
12	RXWAKE	<p>0</p> <p>1</p>	<p>Receiver wakeup detect flag. The SCI sets this bit to indicate that the data currently in SCIRD is an address. RXWAKE is cleared by the following:</p> <ul style="list-style-type: none"> <li>• Setting of the SW nRST bit</li> <li>• Setting of the RESET bit</li> <li>• A system reset</li> <li>• Upon receipt of a data frame.</li> </ul> <p>0 The data in SCIRD is not an address. 1 The data in SCIRD is an address.</p>
11	TX EMPTY	<p>0</p> <p>1</p>	<p>Transmitter empty flag. This flag indicates the transmitter's buffer register(s) (SCITD/TDy) and shift register (SCITXSHF) are empty.</p> <p><b>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.</b></p> <p>0 Transmitter buffer or shift register (or both) are loaded with data. 1 Transmitter buffer and shift registers are both empty.</p>
10	TXWAKE	<p>0</p> <p>1</p> <p>0</p> <p>1</p>	<p>Transmitter wakeup method select. The TXWAKE bit controls whether the data in SCITD should be sent as an address or data frame using multiprocessor communication format. This bit is set to 1 or 0 by software before a byte is written to SCITD and is cleared by the SCI when data is transferred from SCITD to SCITXSHF or by a system reset.</p> <p><b>Note: TXWAKE is not cleared by the SW nRESET bit.</b></p> <p><i>Address-bit mode</i></p> <p>0 Frame to be transmitted will be data (address bit = 0). 1 Frame to be transmitted will be an address (address bit = 1).</p> <p><i>Idle-line mode</i></p> <p>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).</p>

**Table 25-12. SCI Flags Register (SCIFLR) Field Descriptions (continued)**

Bit	Field	Value	Description
9	RXRDY	0 1	<p>Receiver ready flag. The receiver sets this bit to indicate that the SCIRD contains new data and is ready to be read by the CPU or DMA. The SCI generates a receive interrupt when RXRDY flag bit is set if the interrupt-enable bit is set (SCISSETINT[9]); RXRDY is cleared by the following:</p> <ul style="list-style-type: none"> <li>Setting of the SW nRST bit</li> <li>Setting of the RESET bit</li> <li>A system reset</li> <li>Writing a 1 to this bit</li> <li>Reading the SCIRD register in compatibility mode</li> <li>Reading the last data byte RDy of the response in LIN mode</li> </ul> <p><b>Note: The RXRDY flag cannot be cleared by reading the corresponding interrupt offset in the SCIINTVECT0/1 register.</b></p> <p>0 <i>Read:</i> No new data is in SCIRD. <i>Write:</i> Writing a 0 to this bit has no effect.</p> <p>1 <i>Read:</i> New data is ready to be read from SCIRD. <i>Write:</i> The bit is cleared to 0.</p>
8	TXRDY	0 1	<p>Transmitter buffer register ready flag. When set, this bit indicates that the transmit buffer is ready to get another character from a CPU or DMA write.</p> <p>Writing data to SCITD automatically clears this bit. This bit is set after the data of the TX buffer is shifted into the SCITXSHF register. This event can trigger a transmit interrupt after data is copied to the TX shift register SCITXSHF, if the interrupt enable bit TXINT is set.</p> <p><b>Note: 1) TXRDY is also set to 1 either by setting of the RESET bit, enabling SW nRST or by a system reset.</b></p> <p><b>2) The TXRDY flag cannot be cleared by reading the corresponding interrupt offset in the SCIINTVECT0/1 register.</b></p> <p><b>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.</b></p> <p>0 SCITD is full.</p> <p>1 SCITD is ready to receive the next character.</p>
7-4	Reserved	0	Read returns 0. Writes have no effect.
3	BUSY	0 1	<p>Bus busy flag. 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 clears the BUSY bit. If SET WAKEUP INT is set and power down is requested while this bit is set, the SCI automatically prevents lowpower mode from being entered and generates wakeup interrupt. The BUSY bit is controlled directly by the SCI receiver, but this bit can also be cleared by the following:</p> <ul style="list-style-type: none"> <li>Setting the SW nRST bit</li> <li>Setting of the RESET bit</li> <li>A system reset occurring</li> </ul> <p>0 The receiver is not currently receiving a frame.</p> <p>1 The receiver is currently receiving a frame.</p>
2	IDLE	0 1	<p>SCI receiver in idle state. While this bit is set, the SCI looks for an idle period to resynchronize itself with the bit stream. The receiver does not receive any data while the bit is set. The bus must be idle for 11 bit periods to clear this bit. The SCI enters the idle state if one of the following events occurs:</p> <ul style="list-style-type: none"> <li>A system reset</li> <li>An SCI software reset</li> <li>A power down</li> <li>The RX pin is configured as a general I/O pin</li> </ul> <p>0 The idle period has been detected; the SCI is ready to receive.</p> <p>1 The idle period has not been detected; the SCI will not receive any data.</p>

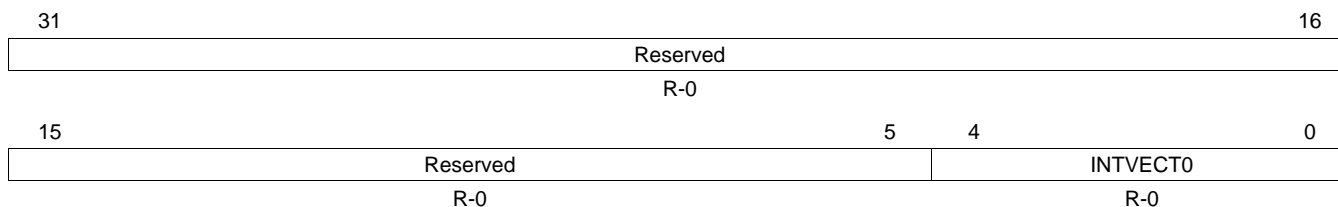
**Table 25-12. SCI Flags Register (SCIFLR) Field Descriptions (continued)**

Bit	Field	Value	Description
1	WAKEUP		<p>Wakeup flag. This bit is set by the SCI when receiver or transmitter activity has taken the module out of power-down mode. An interrupt is generated if the SET WAKEUP INT bit (SCISSETINT[2]) is set. It is cleared by the following:</p> <ul style="list-style-type: none"> <li>• Setting of the SW nRST bit</li> <li>• Setting of the RESET bit</li> <li>• A system reset</li> <li>• Writing a 1 to this bit</li> <li>• Reading the corresponding interrupt offset in SCIINTVECT0/1</li> </ul> <p>For compatibility mode, see the SCI document for more information on lowpower mode.</p> <p>0 <i>Read:</i> The module will not wake up from power-down mode. <i>Write:</i> Writing a 0 to this bit has no effect.</p> <p>1 <i>Read:</i> Wake up from power-down mode. <i>Write:</i> The bit is cleared to 0.</p>
0	BRKDT		<p>SCI break-detect flag. This bit is set when the SCI detects a break condition on the LINRX pin. A break condition occurs when the SCIRX pin remains continuously low for at least 10 bits after a missing first stop bit, that is, after a framing error. Detection of a break condition causes the SCI to generate an error interrupt if the BRKDT INT ENA bit is set. The BRKDT bit is reset by the following:</p> <ul style="list-style-type: none"> <li>• Setting of the SW nRST bit</li> <li>• Setting of the RESET bit</li> <li>• A system reset</li> <li>• Writing a 1 to this bit</li> <li>• Reading the corresponding interrupt offset in SCIINTVECT0/1</li> </ul> <p>0 <i>Read:</i> No break condition has been detected since the last clear. <i>Write:</i> Writing a 0 to this bit has no effect.</p> <p>1 <i>Read:</i> A break condition has been detected. <i>Write:</i> The bit is cleared to 0.</p>

### 25.7.8 SCI Interrupt Vector Offset 0 (SCIINTVECT0)

Figure 25-15 and Table 25-13 illustrate this register.

**Figure 25-15. SCI Interrupt Vector Offset 0 (SCIINTVECT0) [offset = 20h]**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

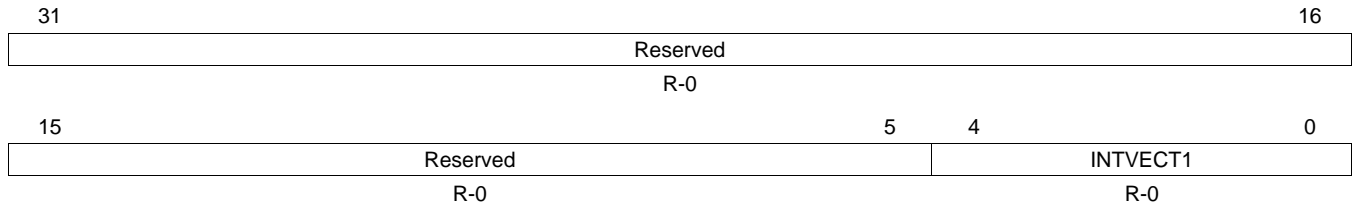
**Table 25-13. SCI Interrupt Vector Offset 0 (SCIINTVECT0) Field Descriptions**

Bit	Field	Value	Description
31-5	Reserved	0	Read returns 0. Writes have no effect.
4-0	INVECT0	0-1Fh	Interrupt vector offset for INT0. This register indicates the offset for interrupt line INT0. A read to this register updates its value to the next highest priority pending interrupt in SCIFLR and clears the flag in SCIFLR corresponding to the offset that was read. See <a href="#">Table 25-1</a> for a list of the interrupts.  <b>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).</b>

**25.7.9 SCI Interrupt Vector Offset 1 (SCIINTVECT1)**

Figure 25-16 and Table 25-14 illustrate this register.

**Figure 25-16. SCI Interrupt Vector Offset 1 (SCIINTVECT1) [offset = 24h]**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

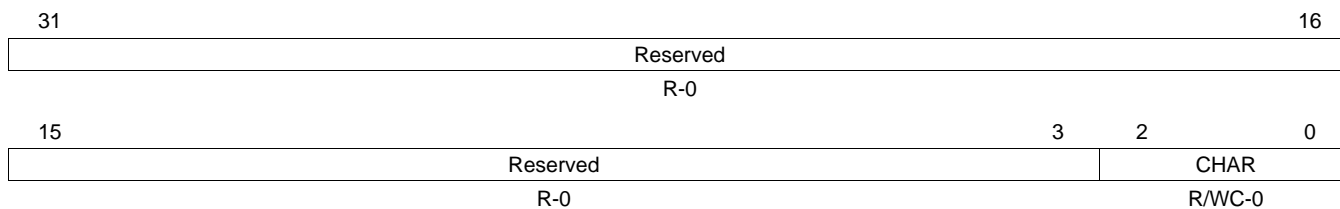
**Table 25-14. SCI Interrupt Vector Offset 1 (SCIINTVECT1) Field Descriptions**

Bit	Field	Value	Description
31-5	Reserved	0	Read returns 0. Writes have no effect.
4-0	INTECT1	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 25-1 for list of interrupts.  <b>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).</b>

### 25.7.10 SCI Format Control Register (SCIFORMAT)

Figure 25-17 and Table 25-15 illustrate this register.

**Figure 25-17. SCI Format Control Register (SCIFORMAT) [offset = 28h]**



LEGEND: R/W = Read/Write; R = Read only; WC = Write in SCI-compatible mode only; -n = value after reset

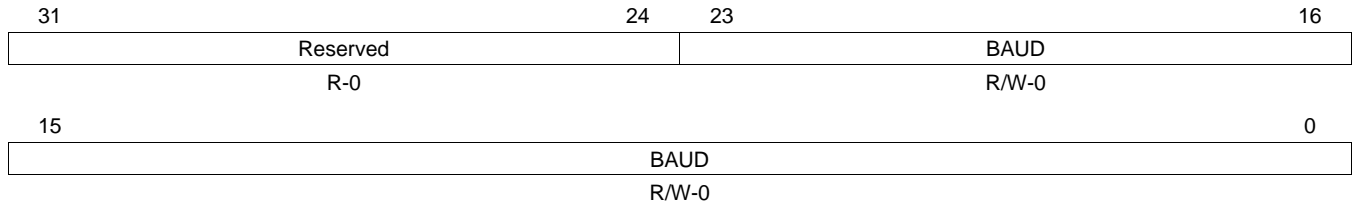
**Table 25-15. SCI Format Control Register (SCIFORMAT) Field Descriptions**

Bit	Field	Value	Description
31-3	Reserved	0	Read returns 0. Writes have no effect.
2-0	CHAR	0-7h	Character length control bits. These bits set the SCI character length from 1 to 8 bits. <b>When data of fewer than eight bits in length is received, it is left justified in SCIRD and padded with trailing zeros. Data read from the SCIRD should be shifted by software to make the received data right justified.</b> <b>Data written to the SCITD should be right justified but does not need to be padded with leading zeros.</b>
		0	The character is 1 bit long.
		1h	The character is 2 bits long.
		2h	The character is 3 bits long.
		3h	The character is 4 bits long.
		4h	The character is 5 bits long.
		5h	The character is 6 bits long.
		6h	The character is 7 bits long.
		7h	The character is 8 bits long.

**25.7.11 Baud Rate Selection Register (BRS)**

This section describes the baud rate selection register. Figure 25-18 and Table 25-16 illustrate this register.

**Figure 25-18. Baud Rate Selection Register (BRS) [offset = 2Ch]**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 25-16. Baud Rate Selection Register (BRS) Field Descriptions**

Bit	Field	Value	Description
31-24	Reserved	0	Read returns 0. Writes have no effect.
23-0	BAUD	0-FF FFFFh	<p>SCI 24-bit baud selection.</p> <p>The SCI has an internally generated serial clock determined by the VCLK and the prescalers BAUD in this register. The SCI uses the 24-bit integer prescaler BAUD value of this register to select one of over 16,700,000.</p> <p>The baud rate can be calculated using the following formulas:</p> $\text{Asynchronous baud value} = \left( \frac{\text{VBUSPLCK Frequency}}{16(\text{Baud} + 1)} \right) \quad (54)$ $\text{Isosynchronous baud value} = \left( \frac{\text{VBUSPLCK Frequency}}{\text{Baud} + 1} \right) \quad (55)$ <p>For BAUD = 0,</p> $\text{Asynchronous baud value} = \left( \frac{\text{VCLK Frequency}}{32} \right) \quad (56)$ $\text{Isosynchronous baud value} = \left( \frac{\text{VCLK Frequency}}{2} \right) \quad (57)$ <p>Table 25-17 contains comparative baud values for different P values, with VCLK = 50 MHz, for asynchronous mode..</p>

**Table 25-17. Comparative Baud Values for Different P Values, Asynchronous Mode <sup>(1)</sup>**

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
- Values are in decimal except for column 2.





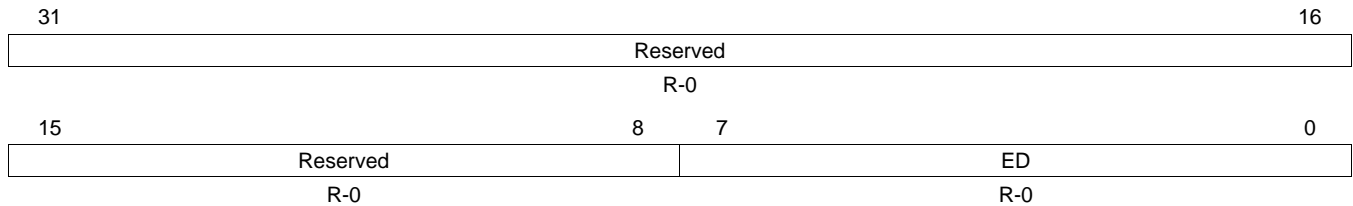
**25.7.12 SCI Data Buffers (SCIED, SCIRD, SCITD)**

The SCI has three addressable registers in which transmit and receive data is stored.

**25.7.12.1 Receiver Emulation Data Buffer (SCIED)**

The SCIED register is addressed at a location different from SCIRD, but is physically the same register. [Figure 25-19](#) and [Table 25-18](#) illustrate this register.

**Figure 25-19. Receiver Emulation Data Buffer (SCIED) [offset = 30h]**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

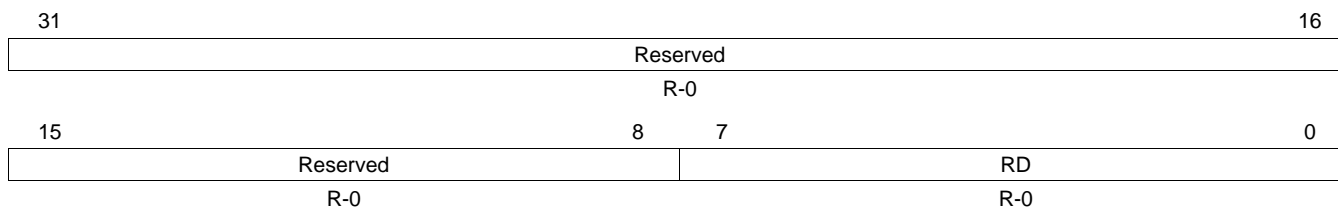
**Table 25-18. Receiver Emulation Data Buffer (SCIED) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Read returns 0. Writes have no effect.
7-0	ED	0-FFh	Emulator data. Reading SCIED[7:0] does not clear the RXRDY flag, unlike reading SCIRD. This register should be used only by an emulator that must continually read the data buffer without affecting the RXRDY flag.

### 25.7.12.2 Receiver Data Buffer (SCIRD)

This register provides a location for the receiver data. [Figure 25-20](#) and [Table 25-19](#) illustrate this register.

**Figure 25-20. Receiver Data Buffer (SCIRD) [offset = 34h]**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

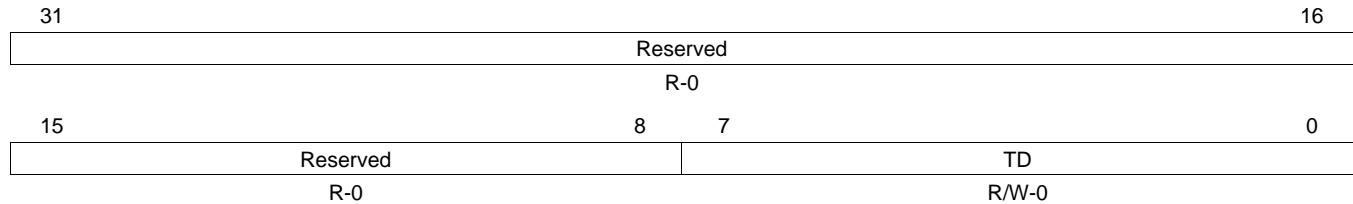
**Table 25-19. Receiver Data Buffer (SCIRD) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Read returns 0. Writes have no effect.
7-0	RD	0-FFh	Receiver data. When a frame has been completely received, the data in the frame is transferred from the receiver shift register SCIRXSHF to this register. As this transfer occurs, the RXRDY flag is set and a receive interrupt is generated if SET RX INT is set. <b>Note: When the data is read from SCIRD, the RXRDY flag is automatically cleared.</b>

**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.

**25.7.12.3 Transmit Data Buffer Register (SCITD)**

Data to be transmitted is written to the SCITD register. [Figure 25-21](#) and [Table 25-20](#) illustrate this register.

**Figure 25-21. Transmit Data Buffer Register (SCITD) [offset = 38h]**


LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 25-20. Transmit Data Buffer Register (SCITD) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Read returns 0. Writes have no effect.
7-0	TD	0-FFh	Transmit data. Data to be transmitted is written to the SCITD register. The transfer of data from this register to the transmit shift register SCITXSHF sets the TXRDY flag, which indicates that SCITD is ready to be loaded with another byte of data. <b>Note: If TX INT ENA is set, this data transfer also causes an interrupt.</b>

---

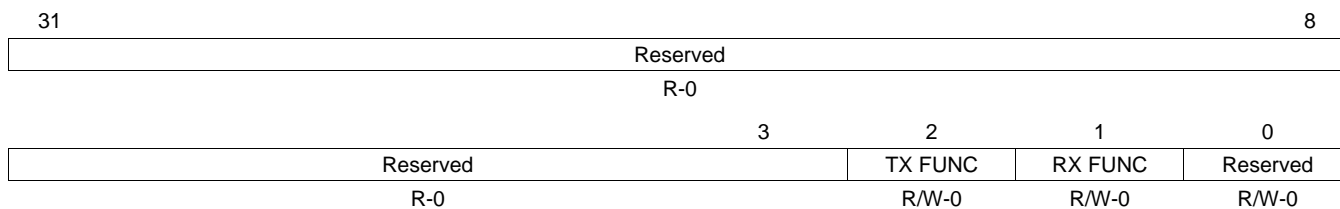
**NOTE:** Data written to the SCITD register that is fewer than eight bits long must be right justified, but it does not need to be padded with leading zeros.

---

### 25.7.13 SCI Pin I/O Control Register 0 (SCIPIO0)

Figure 25-22 and Table 25-21 illustrate this register.

**Figure 25-22. SCI Pin I/O Control Register 0 (SCIPIO0) [offset = 3Ch]**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 25-21. SCI Pin I/O Control Register 0 (SCIPIO0) Field Descriptions**

Bit	Field	Value	Description
31-3	Reserved	0	Read returns 0. Writes have no effect.
2	TX FUNC	0	Transfer function. This bit defines the function of pin SCITX. SCITX is a general-purpose digital I/O pin.
		1	SCITX is the SCI transmit pin.
1	RX FUNC	0	Receive function. This bit defines the function of pin SCIRX. SCIRX is a general-purpose digital I/O pin.
		1	SCIRX is the SCI receive pin.
0	Reserved		Writes have no effect.

### 25.7.14 SCI Pin I/O Control Register 1 (SCIPIO1)

Figure 25-23 and Table 25-22 illustrate this register.

**Figure 25-23. SCI Pin I/O Control Register 1 (SCIPIO1) [offset = 40h]**

31	Reserved				8
R-0					
			3	2	1
Reserved			TX DIR	RX DIR	Reserved
R-0			R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 25-22. SCI Pin I/O Control Register 1 (SCIPIO1) Field Descriptions**

Bit	Field	Value	Description
31-3	Reserved	0	Read returns 0. Writes have no effect.
2	TX DIR	0	Transmit pin direction. This bit determines the data direction on the SCITX pin if it is configured with general-purpose I/O functionality (TX FUNC = 0). See Table 25-23 for the SCITX pin control with this bit and others.
		0	SCITX is a general-purpose input pin.
		1	SCITX is a general-purpose output pin.
1	RX DIR	0	Receive pin direction. This bit determines the data direction on the SCIRX pin if it is configured with general-purpose I/O functionality (RX FUNC = 0). See Table 25-24 for the SCIRX pin control with this bit and others.
		0	SCIRX is a general-purpose input pin.
		1	SCIRX is a general-purpose output pin.
0	Reserved		Writes have no effect.

**Table 25-23. SCITX Pin Control**

Function	TX IN <sup>(1)</sup>	TX OUT	TX FUNC	TX DIR
SCITX	X	X	1	X
General purpose input	X	X	0	0
General purpose output, high	X	1	0	1
General purpose output, low	X	0	0	1

<sup>(1)</sup> TX IN is a read-only bit. Its value always reflects the level of the SCITX pin.

**Table 25-24. SCIRX Pin Control**

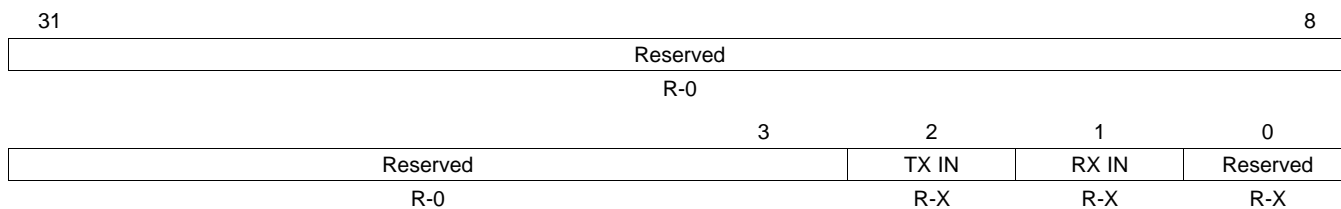
Function	RX IN <sup>(1)</sup>	RX OUT	RX FUNC	RX DIR
SCIRX	X	X	1	X
General purpose input	X	X	0	0
General purpose output, high	X	1	0	1
General purpose output, low	X	0	0	1

<sup>(1)</sup> RX IN is a read-only bit. Its value always reflects the level of the SCIRX pin.

### 25.7.15 SCI Pin I/O Control Register 2 (SCIPIO2)

Figure 25-24 and Table 25-25 illustrate this register.

**Figure 25-24. SCI Pin I/O Control Register 2 (SCIPIO2) [offset = 44h]**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset; -x = Indeterminate

**Table 25-25. SCI Pin I/O Control Register 2 (SCIPIO2) Field Descriptions**

Bit	Field	Value	Description
31-3	Reserved	0	Read returns 0. Writes have no effect.
2	TX IN	0	The SCITX pin is at logic low (0).
		1	The SCITX pin is at logic high (1).
1	RX IN	0	The SCIRX pin is at logic low (0).
		1	The SCIRX pin is at logic high (1).
0	Reserved		Writes have no effect.

### 25.7.16 SCI Pin I/O Control Register 3 (SCIPIO3)

Figure 25-25 and Table 25-26 illustrate this register.

**Figure 25-25. SCI Pin I/O Control Register 3 (SCIPIO3) [offset = 48h]**

31	Reserved				8
	R-0				
		3	2	1	0
	Reserved		TX OUT	RX OUT	Reserved
	R-0		R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 25-26. SCI Pin I/O Control Register 3 (SCIPIO3) Field Descriptions**

Bit	Field	Value	Description
31-3	Reserved	0	Read returns 0. Writes have no effect.
2	TX OUT	0 1	Transmit pin out. This pin specifies the logic to be output on pin SCITX if the following conditions are met: <ul style="list-style-type: none"> <li>• TX FUNC = 0 (SCITX pin is a general-purpose I/O.)</li> <li>• TX DIR = 1 (SCITX pin is a general-purpose output.)</li> </ul> See Table 25-23 for an explanation of this bit's effect in combination with other bits. The output on the SCITX is at logic low (0). The output on the SCITX pin is at logic high (1). (Output voltage is $V_{OH}$ or higher if TXPDR = 0 and output is in high impedance state if TXPDR = 1)
1	RX OUT	0 1	Receive pin out. This bit specifies the logic to be output on pin SCIRX if the following conditions are met: <ul style="list-style-type: none"> <li>• RX FUNC = 0 (SCIRX pin is a general-purpose I/O.)</li> <li>• RX DIR = 1 (SCIRX pin is a general-purpose output.)</li> </ul> See Table 25-24 for an explanation of this bit's effect in combination with the other bits. The output on the SCIRX pin is at logic low (0). The output on the SCIRX pin is at logic high (1). (Output voltage is $V_{OH}$ or higher if RXPDR = 0, and output is in high impedance state if RXPDR = 1)
0	Reserved		Writes have no effect.



### 25.7.17 SCI Pin I/O Control Register 4 (SCIPIO4)

Figure 25-26 and Table 25-27 illustrate this register.

**Figure 25-26. SCI Pin I/O Control Register 4 (SCIPIO4) [offset = 4Ch]**

31	Reserved				8	
R-0						
			3	2	1	0
Reserved			TX SET	RX SET	Reserved	
R-0			R/W-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

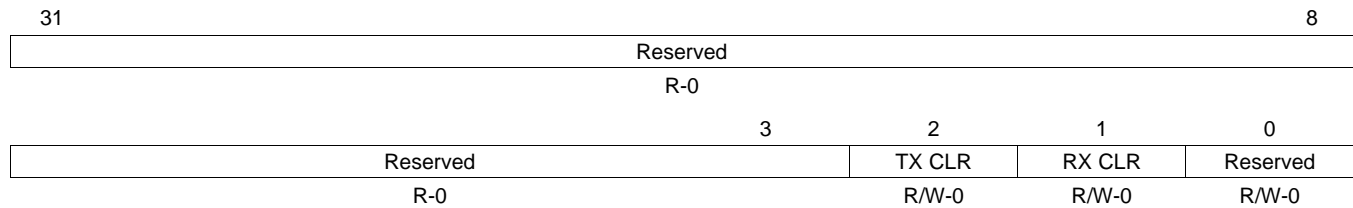
**Table 25-27. SCI Pin I/O Control Register 4 (SCIPIO4) Field Descriptions**

Bit	Field	Value	Description
31-3	Reserved	0	Read returns 0. Writes have no effect.
2	TX SET	0	Transmit pin set. This bit sets the logic to be output on pin SCITX if the following conditions are met: <ul style="list-style-type: none"> <li>TX FUNC = 0 (SCITX pin is a general-purpose I/O.)</li> <li>TX DIR = 1 (SCITX pin is a general-purpose output.)</li> </ul> See Table 25-23 for an explanation of this bit's effect in combination with other bits.
		1	<i>Read and write:</i> The output on SCITX is at logic high (1).
1	RX SET	0	Receive pin set. This bit sets the data to be output on pin SCIRX if the following conditions are met: <ul style="list-style-type: none"> <li>RX FUNC = 0 (SCIRX pin is a general-purpose I/O.)</li> <li>RX DIR = 1 (SCIRX pin is a general-purpose output.)</li> </ul> See Table 25-24 for an explanation of this bit's effect in combination with the other bits.
		1	<i>Read and write:</i> The output on SCIRX is at logic high (1).
0	Reserved		Writes have no effect.

### 25.7.18 SCI Pin I/O Control Register 5 (SCIPIO5)

Figure 25-27 and Table 25-28 illustrate this register.

**Figure 25-27. SCI Pin I/O Control Register 5 (SCIPIO5) [offset = 50h]**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 25-28. SCI Pin I/O Control Register 5 (SCIPIO5) Field Descriptions**

Bit	Field	Value	Description
31-3	Reserved	0	Read returns 0. Writes have no effect.
2	TX CLR	0	Transmit pin clear. This bit clears the logic to be output on pin SCITX if the following conditions are met: <ul style="list-style-type: none"> <li>• TX FUNC = 0 (SCITX pin is a general-purpose I/O.)</li> <li>• TX DIR = 1 (SCITX pin is a general-purpose output.)</li> </ul> <i>Read:</i> The output on SCITX is at logic low (0). <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The output on SCITX is at logic high (1). <i>Write:</i> The output on SCITX is at logic low (0).
1	RX CLR	0	Receive pin clear. This bit clears the logic to be output on pin SCIRX if the following conditions are met: <ul style="list-style-type: none"> <li>• RX FUNC = 0 (SCIRX pin is a general-purpose I/O.)</li> <li>• RX DIR = 1 (SCIRX pin is a general-purpose output.)</li> </ul> <i>Read:</i> The output on SCIRX is at logic low (0). <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> The output on SCIRX is at logic high (1). <i>Write:</i> The output on SCIRX is at logic low (0).
0	Reserved		Writes have no effect.

### 25.7.19 SCI Pin I/O Control Register 6 (SCIPIO6)

Figure 25-28 and Table 25-29 illustrate this register.

**Figure 25-28. SCI Pin I/O Control Register 6 (SCIPIO6) [offset = 54h]**

31	Reserved				8	
R-0						
			3	2	1	0
Reserved			TX FUNC	RX FUNC	Reserved	
R-0			R/W-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 25-29. SCI Pin I/O Control Register 6 (SCIPIO6) Field Descriptions**

Bit	Field	Value	Description
31-3	Reserved	0	Read returns 0. Writes have no effect.
2	TX PDR	0	Transmit pin open drain enable. This bit enables open-drain capability in the output pin SCITX if the following conditions are met: <ul style="list-style-type: none"> <li>TX FUNC = 0 (SCITX pin is a general-purpose I/O.)</li> <li>TX DIR = 1 (SCITX pin is a general-purpose output.)</li> </ul>
		1	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	0	Receive pin open drain enable. This bit enables open-drain capability in the output pin SCIRX if the following conditions are met: <ul style="list-style-type: none"> <li>RX FUNC = 0 (SCIRX pin is a general-purpose I/O.)</li> <li>RX DIR = 1 (SCIRX pin is a general-purpose output.)</li> </ul>
		1	Open drain functionality is disabled; the output voltage is $V_{OL}$ or lower if RXOUT =0 and $V_{OH}$ or higher if RXOUT =1.
		1	Open drain functionality is enabled; the output voltage is $V_{OL}$ or lower if RXOUT =0 and high impedance if RXOUT =1.
0	Reserved		Writes have no effect.

**25.7.20 SCI Pin I/O Control Register 7 (SCIPIO7)**

Figure 25-29 and Table 25-30 illustrate this register.

**Figure 25-29. SCI Pin I/O Control Register 7 (SCIPIO7) [offset = 58h]**

31	Reserved	8
	R-0	
	3	2
	1	0
	Reserved	TX PD
	Reserved	RX PD
	Reserved	Reserved
	R-0	R/W-n
		R/W-n
		R/W-n

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset, Refer to the Terminal Functions in the device datasheet for default pin settings.

**Table 25-30. SCI Pin I/O Control Register 7 (SCIPIO7) Field Descriptions**

Bit	Field	Value	Description
31-3	Reserved	0	Read returns 0. Writes have no effect.
2	TX PD	0	Transmit pin pull control disable. This bit disables pull control capability on the input pin SCITX. The pull control on the SCITX pin is enabled.
		1	The pull control on the SCITX pin is disabled.
1	RX PD	0	Receive pin pull control disable. This bit disables pull control capability on the input pin SCIRX. Pull control on the SCIRX pin is enabled.
		1	Pull control on the SCIRX pin is disabled.
0	Reserved		Writes have no effect.

### 25.7.21 SCI Pin I/O Control Register 8 (SCIPIO8)

Figure 25-30 and Table 25-31 illustrate this register.

**Figure 25-30. SCI Pin I/O Control Register 8 (SCIPIO8) [offset = 5Ch]**

31	Reserved				8	
R-0						
			3	2	1	0
Reserved			TX PSL	RX PSL	Reserved	
R-0			R/W-n	R/W-n	R/W-n	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset, Refer to the Terminal Functions in the device datasheet for default pin settings.

**Table 25-31. SCI Pin I/O Control Register 8 (SCIPIO8) Field Descriptions**

Bit	Field	Value	Description
31-3	Reserved	0	Read returns 0. Writes have no effect.
2	TX PSL	0	The SCITX pin is a pull down.
		1	The SCITX pin is a pull up.
1	RX PSL	0	The SCIRX pin is a pull down.
		1	The SCIRX pin is a pull up.
0	Reserved		Writes have no effect.

### 25.7.22 Input/Output Error Enable (IODFTCTRL) Register

All the bits in the IODFTCTRL register are used in IODFT (I/O design for test) mode only. [Figure 25-31](#) and [Table 25-32](#) illustrate this register. After the basic SCI module configuration, enable the required Error mode to be created followed by IODFT Key enable.

**Figure 25-31. Input/Output Error Enable Register (IODFTCTRL) [offset = 90h]**

31	27	26	25	24
Reserved		FEN	PEN	BRKDTENA
R-0		R/W-0	R/WC-0	R/WC-0
23	21	20	19	18
Reserved		PIN SAMPLE MASK		TX SHIFT
R-0		R/W-0		R/W-0
15	12	11	8	
Reserved		IODFTENA		
R-0		R/WP-0	R/WP-1	R/WP-0
R/WP-1				
7	2		1	0
Reserved			LPB ENA	RXPENA
R-0			R/WP-0	R/WP-0

LEGEND: R/W = Read/Write; R = Read only; WC = Write in sci-compatible mode only; WP = Write in privilege mode only; -n = value after reset

**NOTE:**

1. All the bits are used in IODFT mode only.
2. Each IODFT are expected to be checked individually.

**Table 25-32. Input/Output Error Enable Register (IODFTCTRL) Field Descriptions**

Bit	Field	Value	Description
31-27	Reserved	0	Read returns 0. Writes have no effect.
26	FEN	0	Frame error enable. This bit is used to create a frame error. No error is created.
		1	The stop bit received is ANDed with 0 and passed to the stop bit check circuitry.
25	PEN	0	Parity error enable. This bit is used to create a parity error. No parity error occurs.
		1	The parity bit received is toggled so that a parity error occurs.
24	BRKD TENA	0	Break detect error enable. This bit is used to create a BRKDT error. 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 <sub>BITS</sub> so that a BRKDT error occurs.
32-21	Reserved	0	Read returns 0. Writes have no effect.
20-19	PIN SAMPLE MASK	0	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. No mask is used.
		1h	Invert the TX Pin value at 7th.SCLK.
		2h	Invert the TX Pin value at 8th.SCLK.
		3h	Invert the TX Pin value at 9th.SCLK.

**Table 25-32. Input/Output Error Enable Register (IODFTCTRL) Field Descriptions (continued)**

Bit	Field	Value	Description
18-16	TX SHIFT	0 1h 2h 3h 4h 5h 6h 7h	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.  No delay occurs. The value is delayed by 1 SCLK. The value is delayed by 2 SCLK. The value is delayed by 3 SCLK. The value is delayed by 4 SCLK. The value is delayed by 5 SCLK. The value is delayed by 6 SCLK. No delay occurs.
15-12	Reserved	0	Read returns 0. Writes have no effect.
11-8	IODFTENA	Ah All Others	IODFT enable key. Write access permitted in Priviledge mode only.  IODFT is enabled. IODFT is disabled.
7-2	Reserved	0	Read returns 0. Writes have no effect.
1	LPBENA	0 1	Module loopback enable. Write access permitted in Priviledge mode only.  <b>Note: In analog loopback mode the complete communication path through the I/Os can be tested, whereas in digital loopback mode the I/O buffers are excluded from this path.</b>  Digital loopback is enabled. Analog loopback is enabled in module I/O DFT mode when IODFTENA = 1010.
0	RXPENA	0 1	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 loopback mode)  Analog loopback through the transmit pin is enabled. Analog loopback through the receive pin is enabled.

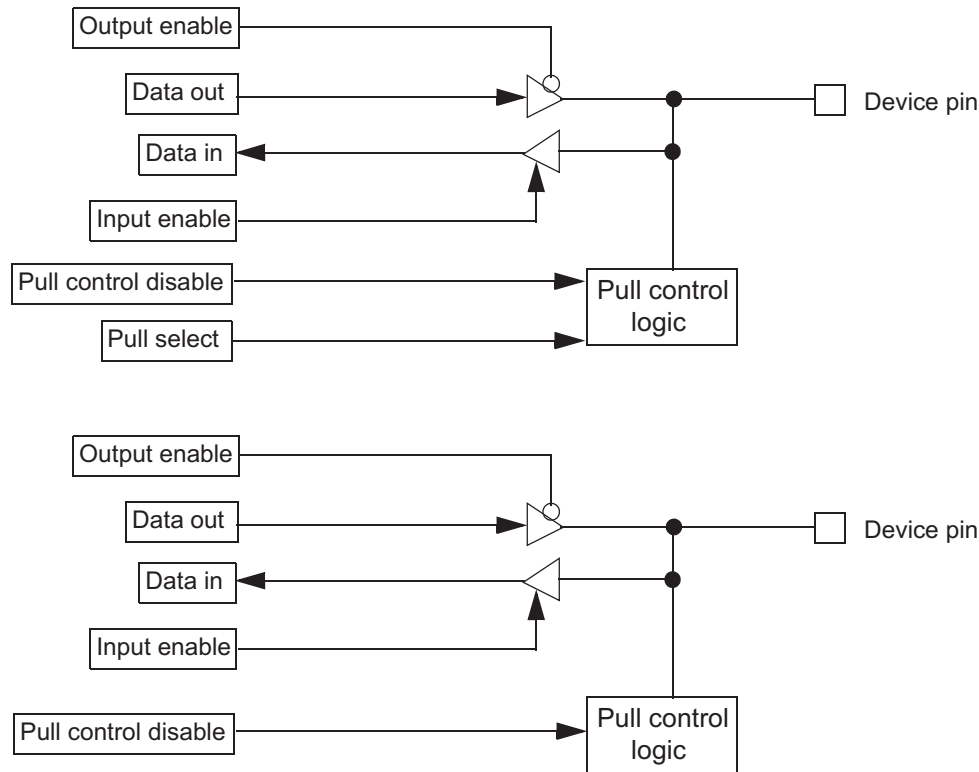
## 25.8 GPIO Functionality

The following sections apply to all device pins that can be configured as functional or general-purpose I/O pins.

### 25.8.1 GPIO Functionality

Figure 25-32 illustrates the GPIO functionality.

Figure 25-32. GPIO Functionality



### 25.8.2 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. Input buffer is enabled.
- Output buffer. The output buffer is disabled.

### 25.8.3 Out of Reset

The following apply if the device is out of reset:

- Pull control. The pull control is enabled by clearing the PD (pull control disable) bit in the SCIO7 register (Section 25.7.20). In this case, if the PSL (pull select) bit in the SCIO8 register (Section 25.7.21) is set, the pin will have a pull-up. If the PSL bit is cleared, the pin will have a pull-down. If the PD bit is set in the control register, there is no pull-up or pull-down on the pin.
- Input buffer. The input buffer is always enabled in functional mode.



**NOTE:** The pull-disable logic depends on the pin direction. It is independent of whether the device is in I/O or functional mode. If the pin is configured as output or transmit, then the pulls are disabled automatically. If the pin is configured as input or receive, the pulls are enabled or disabled depending on bit PD in the pull disable register SCIPIO7 ([Section 25.7.20](#)).

- Output buffer. A pin can be driven as an output pin if the TX DIR bit is set in the pin direction control register (SCIPIO1; [Section 25.7.14](#)) AND the open-drain feature is not enabled in the SCIPIO6 register ([Section 25.7.19](#)).

#### 25.8.4 Open-Drain Feature Enabled on a Pin

The following apply if the open-drain feature is enabled on a pin.

- The output buffer is enabled if a low signal is being driven on to the pin.
- The output buffer is disabled (the direction control signal DIR is internally forced low) if a high signal is being driven on to the pin.

**NOTE:** The open-drain feature is available only in I/O mode (SCIPIO0; [Section 25.7.13](#)).

#### 25.8.5 Summary

The behavior of the input buffer, output buffer, and the pull control is summarized in [Table 25-33](#).

**Table 25-33. Input Buffer, Output Buffer, and Pull Control Behavior as GPIO Pins <sup>(1)</sup>**

Device under Reset?	Pin Direction (DIR) <sup>(2)</sup>	Pull Disable (PULDIS) <sup>(3)</sup>	Pull Select (PULSEL) <sup>(4)</sup>	Pull Control	Output Buffer	Input Buffer
Yes	X	X	X	Device- and module-specific	Disabled	Enabled
No	0	0	0	Pull down	Disabled	Enabled
No	0	0	1	Pull up	Disabled	Enabled
No	0	1	0	Disabled	Disabled	Enabled
No	0	1	1	Disabled	Disabled	Enabled
No	1	X	X	Disabled	Enabled	Enabled

<sup>(1)</sup> X = Don't care

<sup>(2)</sup> DIR = 0 for input, 1 for output

<sup>(3)</sup> PULDIS = 0 for enabling pull control  
= 1 for disabling pull control

<sup>(4)</sup> PULSEL = 0 for pull-down functionality  
= 1 for pull-up functionality

## ***Inter-Integrated Circuit (I2C) Module***

This chapter describes the inter-integrated circuit (I2C or I<sup>2</sup>C) module. The I2C is a multi-master communication module providing an interface between the Texas Instruments (TI) microcontroller and devices compliant with Philips Semiconductor I<sup>2</sup>C-bus specification version 2.1 and connected by an I2C-bus. This module will support any slave or master I2C compatible device.

Topic	Page
<b>26.1 Overview .....</b>	<b>1504</b>
<b>26.2 I2C Module Operation .....</b>	<b>1509</b>
<b>26.3 I2C Operation Modes .....</b>	<b>1513</b>
<b>26.4 I2C Module Integrity .....</b>	<b>1515</b>
<b>26.5 Operational Information .....</b>	<b>1517</b>
<b>26.6 I2C Registers .....</b>	<b>1520</b>
<b>26.7 Sample Waveforms .....</b>	<b>1551</b>

## 26.1 Overview

The I2C has the following features:

- Compliance to the Philips I<sup>2</sup>C bus specification, v2.1 (*The I2C Specification*, Philips document number 9398 393 40011)
  - Bit/Byte format transfer
  - 7-bit and 10-bit device addressing modes
  - General call
  - START byte
  - Multi-master transmitter/ slave receiver mode
  - Multi-master receiver/ slave transmitter mode
  - Combined master transmit/receive and receive/transmit mode
  - Transfer rates of 10 kbps up to 400 kbps (Phillips fast-mode rate)
- Free data format
- Two DMA events (transmit and receive)
- DMA event enable/disable capability
- Seven interrupts that can be used by the CPU
- Operates with VBUS frequency from 6.7 MHz up
- Operates with module frequency between 6.7 MHz to 13.3 MHz
- Module enable/disable capability
- The SDA and SCL are optionally configurable as general purpose I/O
- Slew rate control of the outputs
- Open drain control of the outputs
- Programmable pullup/pulldown capability on the inputs
- Supports Ignore NACK mode

---

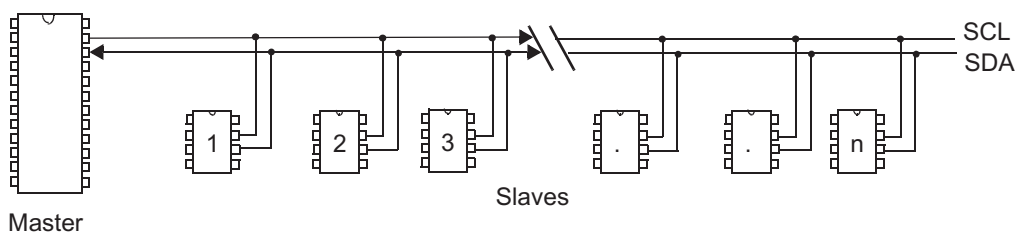
**NOTE:** This I2C module does **not** support:

- High-speed (HS) mode
  - C-bus compatibility mode
  - The combined format in 10-bit address mode (the I2C sends the slave address second byte every time it sends the slave address first byte)
- 

### 26.1.1 Introduction to the I2C Module

The I2C module supports any slave or master I2C-compatible device. [Figure 26-1](#) shows an example of multiple I2C serial ports connected for a two-way transfer from one device to another device.

**Figure 26-1. Multiple I2C Modules Connection Diagram**



## 26.1.2 Functional Overview

The I2C module is a serial bus that supports multiple master devices. In multimaster mode, one or more devices can be connected to the same bus and are capable of controlling the bus. Each I2C device on the bus is recognized by a unique address and can operate as either a transmitter or a receiver, depending on the function of the device. In addition to being a transmitter or receiver, a device connected to the I2C bus can also be considered a master or a slave when performing data transfers.

---

**NOTE:** A master device is the device that initiates the data transfer on a bus and generates the clock signal that permits the transfer. During the transmission, any device addressed by the master is considered the slave.

---

Data is communicated to devices interfacing to the I2C module using the serial data pin (SDA) and the serial clock pin (SCL) as shown in [Figure 26-2](#). These two wires carry information between the device and the other devices connected to the I2C bus. Both SDA and SCL pins on the device are bidirectional. They must be connected to a positive supply voltage through a pull-up resistor. When the bus is free, both pins are high. The driver of these two pins has an open-drain configuration to perform the wired-AND function.

The device has a special mode that can be entered to ignore a NACK generated from non-compliant I2C devices that are incapable of generating an ACK.

The I2C module consists of the following primary blocks:

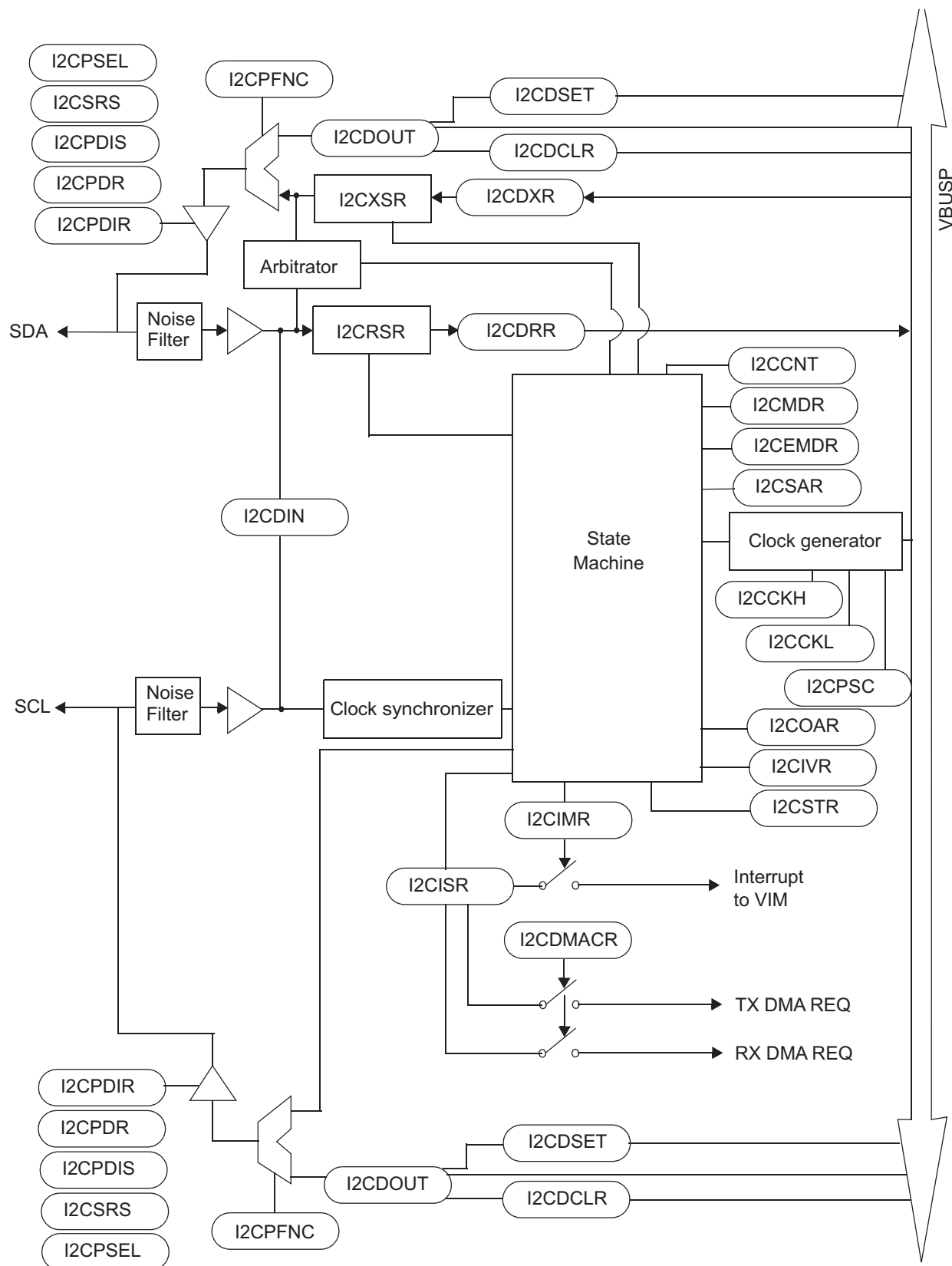
- A serial Interface: one data pin (SDA) and one clock pin (SCL)
- The device register interface
  - Data registers to temporarily hold received data and transmitted data traveling between the SDA pin and the CPU or the DMA
  - Control and status registers
- A prescaler to divide down the input clock that is driven to the I2C module
- A peripheral bus interface to enable the CPU and DMA to access the I2C module registers
- An arbitrator to handle arbitration between the I2C module (when configured as a master) and another master
- Interrupt generation logic (interrupts can be sent to the CPU)
- A clock synchronizer that synchronizes the I2C input clock (from the system module) and the clock on the SCL pin, and synchronizes data transfers with masters of different clock speeds.
- A noise filter on each of the two serial pins
- DMA event generation logic that synchronizes data reception and data transmission in the I2C module for DMA transmission

In [Figure 26-2](#), the CPU or the DMA writes data for transmission to I2CDXR and reads received data from I2CDRR. When the I2C module is configured as a transmitter, data written to I2CDXR is copied to I2CXSR and shifted out one bit at a time. When the I2C module is configured as a receiver, received data is shifted into I2CRSR and then copied to I2CDRR.

When the I2C function is not needed, the pins may be controlled as general-purpose input/output (GPIO) pins. The I/O structure of each pin includes:

- programmable slew rate control of the outputs
- open drain mode
- programmable pull enable/disable on the input
- programmable pull up/pull down function on the input

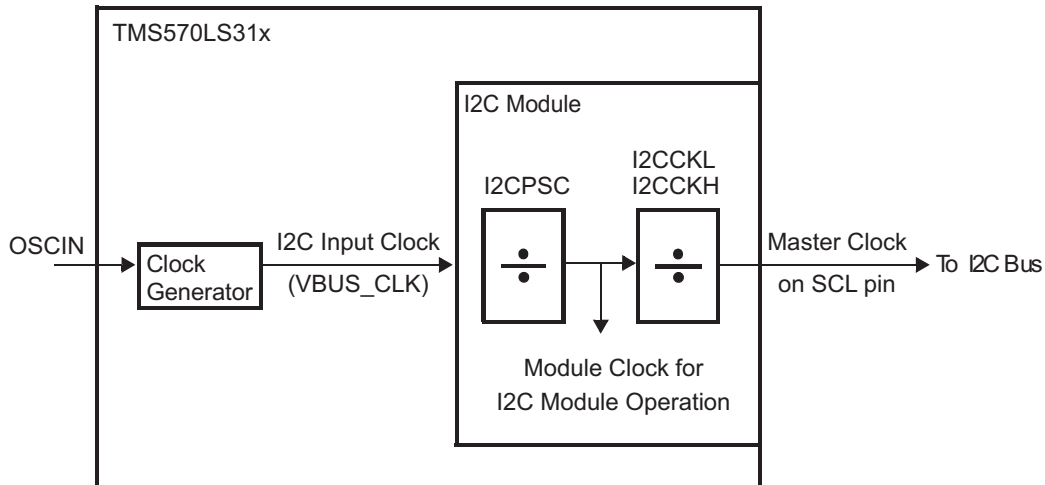
Figure 26-2. Simple I2C Block Diagram



### 26.1.3 Clock Generation

As shown in Figure 26-3, the I2C module uses the input clock generated from the device clock generator to generate the module clock and master clock. The I2C input clock is the device peripheral clock (VBUS\_CLK). The clock is then divided twice more inside the I2C module to produce the module clock and the master clock.

Figure 26-3. Clocking Diagram for the I2C Module



The module clock determines the frequency at which the I2C module operates. A programmable prescaler in the I2C module divides down the input clock to produce the module clock. To specify the divide-down value, initialize the I2CPSC field of the prescaler register, I2CPSC. The resulting frequency is:

$$ModuleClockFrequency = \frac{I2CInputClockFrequency}{(I2CPSC + 1)} \quad (58)$$

The module clock frequency must be between 6.7MHz and 13.3MHz. The prescaler can only be initialized while the I2C module is in the reset state (IRS = 0 in I2CMDR). The prescaled frequency takes effect only when IRS is changed to 1. Changing the I2CPSC value while IRS = 1 has no effect.

The master clock appears on the SCL pin when the I2C module is configured to be a master on the I2C bus. This clock controls the timing of the communication between the I2C module and a slave. As shown in Figure 26-3, a second clock divider in the I2C module divides down the module clock to produce the master clock. The clock divider uses the I2CCKL to divide down the low portion of the module clock signal and uses the I2CCKH to divide down the high portion of the module clock signal.

The resulting frequency is:

$$MasterClockFrequency = \frac{ModuleClockFrequency}{(I2CCKL + d) + (I2CCKH + d)} \quad (59)$$

$$MasterClockFrequency = \frac{I2CInputClockFrequency}{(I2CPSC + 1)((I2CCKL + d) + (I2CCKH + d))} \quad (60)$$

where *d* depends on the value of I2CPSC:

I2CPSC	d
0	7
1	6
Greater than 1	5

**NOTE:** The master clock frequency defined above does not include rise/fall time and latency of the synchronizer inside the module. The actual transfer rate will be slower than the value calculated from the formula above. Also, due to the nature of SCL synchronization, the SCL clock period could change if SCL synchronization is taking place.

---

## 26.2 I2C Module Operation

The following section discusses how the I2C module operates.

### 26.2.1 Input and Output Voltage Levels

One clock pulse is generated by the master device for each data bit transferred. Because of a variety of different technology devices that can be connected to the I2C-bus, the levels of logic 0 (low) and logic 1 (high) are not fixed and depend on the associated level of  $V_{CCIO}$ . For details, see the device specific data sheet.

### 26.2.2 I2C Module Reset Conditions

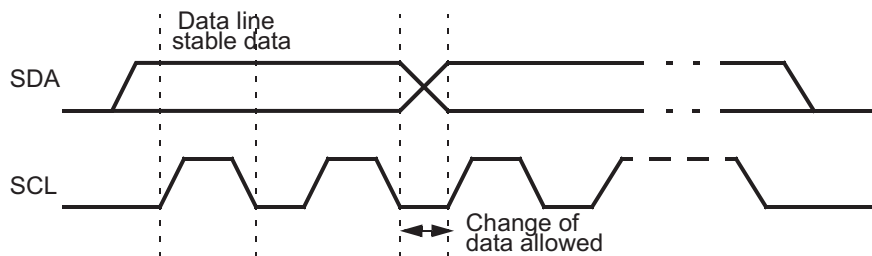
The I2C module can be reset in the following two ways:

- Through the global peripheral reset. A device reset causes a global peripheral reset.
- By clearing the  $\overline{IRS}$  bit in the I2C mode register (I2CMDR). When the global peripheral reset is removed, the  $\overline{IRS}$  bit is cleared to 0, keeping the I2C module in the reset state.

### 26.2.3 I2C Module Data Validity

The data on the SDA must be stable during the high period of the clock. See Figure 26-4. The high and low state of the data line, the SDA, can only change when the clock signal is low.

Figure 26-4. Bit Transfer on the I2C Bus

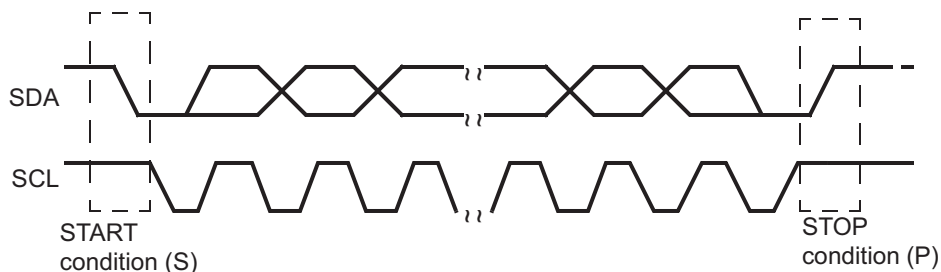


### 26.2.4 I2C Module Start and Stop Conditions

START and STOP conditions are generated by a master I2C module.

- The START condition is defined as a high-to-low transition on the SDA line while SCL is high. A master drives this condition to indicate the start of data transfer. The bus is considered to be busy after the START condition, and the bus busy bit (BB) in I2CSR is set to 1.
- The STOP condition is defined as a low-to-high transition on the SDA line while SCL is high. A master drives this condition to indicate the end of data transfer. The bus is considered to be free after the STOP condition, therefore the BB bit in I2CSR is cleared to 0.

Figure 26-5. I2C Module START and STOP Conditions



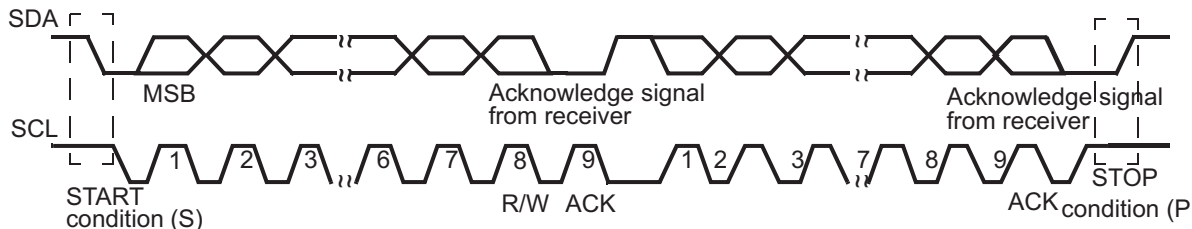
For the I2C module to start a data transfer with a START condition, the master mode bit (MST) and the START condition bit (STT) in the I2CMDR must both be set to 1. For the I2C module to end a data transfer with a STOP condition, the STOP condition bit (STP) must be set to 1. When the BB bit is set to 1 and the STT bit is set to 1, a repeated START condition is generated.



## 26.2.5 Serial Data Formats

The I2C module operates in byte data format. Each message put on the SDA line is 2 to 8-bits long. The number of messages that can be transmitted or received is unrestricted. The data is transferred with the most significant bit (MSB) first (Figure 26-6). Each message is followed by an acknowledge bit from the I2C if it is in receiver mode. The I2C module does not support little endian systems.

**Figure 26-6. I2C Module Data Transfer**



The first byte after a START condition (S) always consists of 8 bits that comprise either a 7-bit address plus the R/W bit, or 8 data bits. The eighth bit, R/W, in the first byte determines the direction of the data. When the R/W bit is 0, the master writes (transmits) data to a selected slave device; when the R/W bit is 1, the master reads (receives) data from the slave device. In acknowledge mode, an extra bit dedicated for the acknowledgement (ACK) bit is inserted after each message.

The I2C module supports the following formats:

- 7-bit addressing format (Figure 26-7)
- 10-bit addressing format (Figure 26-8)
- 7-bit/10-bit addressing format with repeated START condition (Figure 26-9)
- Free-data format (Figure 26-10)

### 26.2.5.1 7-Bit Addressing Format

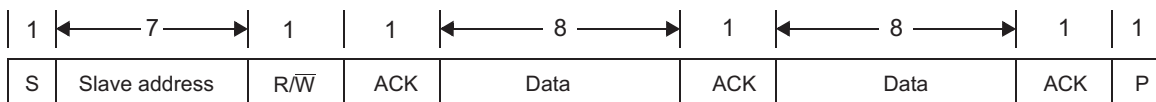
In the 7-bit addressing format (Figure 26-7), the first byte after the START condition consists of a 7-bit slave address followed by the R/W bit (in the LSB). The R/W bit determines the direction of the data transfer:

- R/W = 0: The master writes (transmits) data to the addressed slave.
- R/W = 1: The master reads (receives) data from the slave.

An extra clock cycle dedicated for acknowledgement (ACK) is inserted after each byte. If the ACK is inserted by the slave after the first byte from the master, it is followed by n bits of data from the transmitter (master or slave, depending on the R/W bit). The device I2C allows n to be a number between 2 to 8, programmable by the bit count (BC) field of I2CMDR. After the data bits have been transferred, the receiver inserts an ACK bit.

To select the 7-bit addressing format, write 0 to the expanded address enable (XA) bit of I2CMDR and make sure the free data format mode is off (FDF = 0 in I2CMDR).

**Figure 26-7. I2C Module 7-Bit Addressing Format**



### 26.2.5.2 10-Bit Addressing Format

The 10-bit addressing format is similar to the 7-bit addressing format, but the master sends the slave address in two separate byte transfers. In the 10-bit addressing format (Figure 26-8), the first byte is 11110b, the two MSBs of the 10-bit slave address, and the R/W bit. The ACK bit is inserted after each byte. The second byte is the remaining 8 bits of the 10-bit slave address. The slave must send an acknowledgement after each of the two byte transfers. Once the master has written the second byte to the slave, the master can either write data or use repeated a START condition to change the data direction.

To select the 10-bit addressing format, write 1 to the expanded address enable (XA) bit of I2CMDR and make sure the free data format mode is off (FDF = 0 in I2CMDR).

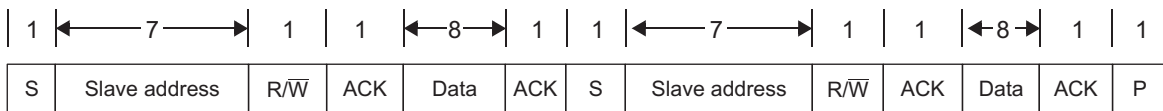
**Figure 26-8. I2C Module 10-bit Addressing Format**



**26.2.5.3 Using the Repeated START Condition**

At the end of each byte, the master can drive another START condition (Figure 26-9). Using this capability, a master can transmit/receive any number of data bytes before generating a STOP condition. The length of a data byte can be from 2 to 8 bits. The repeated START condition can be used with the 7-bit addressing, 10-bit addressing, or the free data formats.

**Figure 26-9. I2C Module 7-Bit Addressing Format with Repeated START**

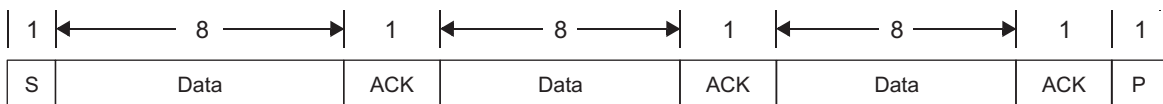


**26.2.5.4 Free Data Format**

In this format (Figure 26-10), the first byte after a START condition is a data byte. The ACK bit is inserted after each byte, followed by another 8 bits of data. No address or data direction bit is sent. Therefore, the transmitter and receiver must both support the free data format. The direction of data transmission (transmit or receive) remains constant throughout the transfer.

To select the free data format, write a 1 to the free data format (FDF) bit of the I2CMDR. The free data format is not supported in the digital loop back mode.

**Figure 26-10. I2C Module in Free Data Format**



**26.2.6 NACK Bit Generation**

When the I2C module is a receiver (master or slave), it can acknowledge or ignore bits sent by the transmitter. To ignore any new bits, the I2C module must send a no-acknowledge (NACK) bit during the acknowledge cycle on the bus. Table 26-1 summarizes the various ways a NACK can be generated.

**Table 26-1. Ways to Generate a NACK Bit**

I2C Module Condition	Basic NACK Bit Generation Options	Additional Option
Slave receiver mode	Disable data transfers (STT = 0)Allow an overrun condition (RSFULL = 1)Reset the module (IRS = 0)	Set the NACKMOD bit before the rising edge of the last data bit you intend to receive.
Master receiver mode and repeat mode (RM = 1)	Generate a STOP condition (STP = 1)Reset the module (IRS = 0)	Set the NACKMOD bit before the rising edge of the last data bit you intend to receive.
Master receiver mode with non-repeat mode (RM = 0)	If STP = 1, allow the internal data counter to count down to 0 and thus force a STOP condition.If STP = 0, make STP = 1 to generate a STOP condition.Reset the module (IRS = 0)	Set the NACKMOD bit before the rising edge of the last data bit you intend to receive.

---

In some applications, the slave cannot generate the ACK signal. If the IGNACK bit is set in the I2CEMDR register, the resulting NACK will be ignored and the I2C block will continue the data transfer.

## 26.3 I2C Operation Modes

### 26.3.1 Master Transmitter Mode

All masters begin in this mode. The I2C module is a master and transmits control information and data to a slave. In this mode, data assembled in any of the addressing formats shown in [Figure 26-7](#), [Figure 26-8](#), or [Figure 26-9](#) is shifted out onto the SDA pin and synchronized with the self-generated clock pulses on the SCL pin. The clock pulses are inhibited and the SCL pin is held low when the intervention of the device is required ( $\overline{XSMT} = 0$ ) after a byte has been transmitted.

---

**NOTE:** If the I2C is configured for two simultaneous master transmissions, wait until the MST and BB have been reset before performing the second master transmission.

---

Failure to wait for the MST and BB to reset will prevent the start condition on the second transfer from being issued and the bus BB will not be set. Typically the end of the first transfer is handled by polling BB. However, the MST bit is not reset at the same instant as the BB bit. As a result, when the second master transmission is initiated before the resetting of the MST, the MST bit for the second transfer is reset. This prevents the I2C from recognizing itself as the master, thus failing to occupy the bus.

### 26.3.2 Master Receiver Mode

In this mode, the I2C module is a master and receives data from a slave. This mode can only be entered from the master transmitter mode (the I2C module must first transmit a command to the slave). In any of the addressing formats shown in [Figure 26-7](#), [Figure 26-8](#), or [Figure 26-9](#), the master receiver mode is entered after the slave address byte and the R/W bit have been transmitted (if the R/W bit is 1). Serial data bits received on the SDA pin are shifted in with the self-generated clock pulses on the SCL pin. The clock pulses are inhibited and the SCL is held low when the intervention of the device is required ( $RSFULL = 1$ ) after a byte has been received. At the end of the transfer, the master-receiver signals the end of data to the slave-transmitter by not generating an acknowledge on the last byte that was clocked out of the slave. The slave-transmitter then releases the data line allowing the master-receiver to generate a STOP condition or a repeated START condition.

In many applications, the size of the message is in the initial bytes of the message itself. Since the size of the message is not known to the master before the transmission/reception starts, the master must use the repeat mode in order to force the stop condition when the reception is completed. The repeat mode is enabled by setting the RM bit to 1. Due to the double buffer implementation on the receive side, the master must generate the stop condition ( $STP = 1$ ) after reading the (message size - 1)<sup>th</sup> data.

### 26.3.3 Slave Transmitter Mode

In this mode, the I2C module is a slave and transmits data to a master. This mode can only be entered from the slave receiver mode (The I2C module must first receive a command from the master). In any of the addressing formats shown in [Figure 26-7](#), [Figure 26-8](#), or [Figure 26-9](#), the slave transmitter mode is entered if the slave address byte is the same as its own address and the R/W bit has been transmitted (if the R/W bit is set to 1). The slave transmitter shifts the serial data out on the SDA pin with the clock pulses that are generated by the master device. The slave device does not generate the clock, but it can hold the SCL pin low when intervention of the device is required ( $\overline{XSMT} = 0$ ) after a byte has been transmitted.

### 26.3.4 Slave Receiver Mode

In this mode, the I2C module is a slave and receives data from a master. All slaves begin in this mode. Serial data bits received on the SDA pin are shifted in with the clock pulses that are generated by the master device. The slave device does not generate the clock, but it can hold the SCL pin low while intervention of the device is required ( $RSFULL = 1$ ) after a byte has been received.

### **26.3.5 Low Power Mode**

The I2C module can be placed in low-power mode by a global low-power mode initiated by the system (by writing to the Peripheral Power-Down Set Register in the Peripheral Central Resource (PCR) module; Refer to the device Architecture Specification).

In effect, low-power mode shuts down all the clocks to the module. In global low-power mode, no registers are visible to the software; nothing can be written to or read from any register.

### **26.3.6 Free Run Mode**

The I2C module can be placed in free run mode when the FREE bit (I2CMDR.14) is set to 1. This bit is primarily used on an emulator when encountering a breakpoint while debugging software. When the FREE bit is set to 0, the I2C responds differently depending on whether the SCL is high or low. If the SCL is low, the I2C stops immediately and keeps driving the SCL low whether the I2C is the master transmitter or receiver. If the SCL is high, the I2C waits until the SCL becomes a low and then stops. If the I2C is a slave, it stops when the transmission/reception completes.

### **26.3.7 Ignore NACK Mode**

The I2C module can be placed in the ignore NACK mode by setting the IGNACK bit in the I2CEMDR register. This mode allows an I2C module that is configured as a master transmitter to ignore a NACK from a slave device that is not capable of generating a proper ACK signal.

## 26.4 I2C Module Integrity

The following section discusses how the I2C module maintains priorities and order among signals and commands.

### 26.4.1 Arbitration

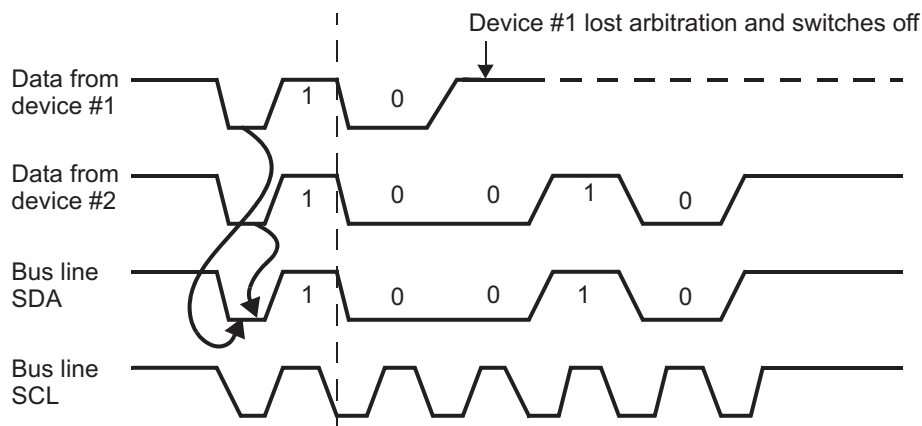
If two or more master transmitters simultaneously start a transmission on the same bus, an arbitration procedure is invoked. Figure 26-11 illustrates the arbitration procedure between two devices. The arbitration procedure uses the data presented on the SDA bus by the competing transmitters. The first master transmitter that generates a high is overruled by the other master that generates a low. The arbitration procedure gives priority to the device that transmits the serial data stream with the lowest binary value. The master transmitter that loses the arbitration switches to the slave receiver mode, sets the arbitration lost (AL) flag, and generates the arbitration-lost interrupt. The data transmitted by the other master module is salvaged, and the I2C continues to receive data from the master module. Should two or more devices send identical first bytes, arbitration continues on the subsequent bytes.

If, during a serial transfer, the arbitration procedure is still in progress when a repeated START condition or STOP condition is transmitted to I2C bus, the master transmitters involved must send the repeated START condition or STOP condition at the same position in the format frame. In other words, arbitration is not allowed between:

- A repeated START condition and a data bit
- A STOP condition and a data bit
- A repeated START condition and a STOP condition

Slaves are not involved in the arbitration procedure.

**Figure 26-11. Arbitration Procedure Between Two Master Transmitters**



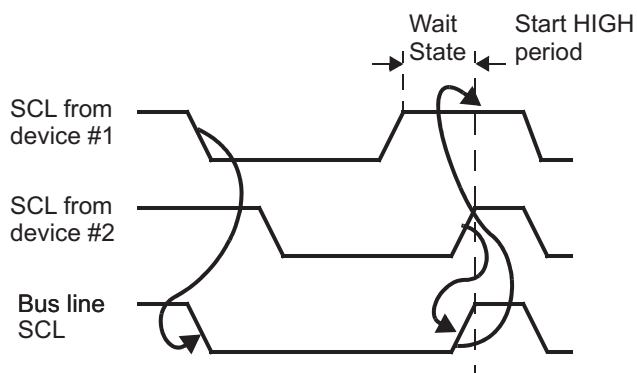
### 26.4.2 I2C Clock Generation and Synchronization

Under normal conditions only one master device generates the clock signal; the SCL. During the arbitration procedure, however, there are two or more master devices and the clock must be synchronized so that the data output can be compared. Figure 26-12 illustrates clock synchronization. The wired-AND property of the SCL line means that a device that first generates a low period on the SCL overrules the other devices. At this high-to-low transition, the clock generators of the other devices are forced to start their own low period. The SCL line is held low by the device with the longest low period. The other devices that finish their low periods must wait for the SCL line to be released before starting their high periods. A synchronized signal on the SCL is obtained where the slowest device determines the length of the low period and the fastest device determines the length of the high period.

If a device pulls down the clock line for a longer time, the result is that all clock generators must enter the wait state. In this way, a slave slows down a fast master and the slow device creates enough time to store a received byte or to prepare a byte to be transmitted.

**NOTE: I2C Protocol Fault**

The following conditions violate the clock spec as defined in the Philips I<sup>2</sup>C bus specification, v2.1 (*The I2C Specification*, Philips document number 9398 393 40011), and will result in an I2C protocol fault:  $I2CCLKH = 2$   $I2CCLKL = 2I2CPSC = 2$  This will cause the SDA data transition to occur while the SCL is high.

**Figure 26-12. Synchronization of Two I2C Clock Generators During Arbitration****26.4.3 Prescaler**

The I2C module is operated by the module clock. This clock is generated by way of the I2C prescaler block. The prescaler block consists of a 8-bit register, I2CPSC, used for dividing down the device peripheral clock (VBUS\_CLK) to obtain a module clock between 6.7 MHz and 13.3 MHz.

**26.4.4 Noise Filter**

The noise filter is used to suppress any noises that are 50ns or less. It is designed to suppress noise with one module clock, assuming the lower and upper limits of the module clock are 6.7MHz and 13.3MHz, respectively.

## 26.5 Operational Information

The following section provides specific information about how the I2C module operates.

### 26.5.1 I2C Module Interrupts

The I2C module generates seven types of interrupts. These seven interrupts are accompanied with seven interrupt mask bits in the interrupt mask register (I2CIMR) and with seven interrupt flag bits in the status register (I2CSR).

#### 26.5.1.1 I2C Interrupt Requests

The I2C module generates the interrupt requests described below. All requests are multiplexed through an arbiter into a single I2C interrupt request to the CPU. Each interrupt request has a flag bit and an enable bit. Interrupts must be enabled prior to the occurrence of the expected interrupt condition. When one of the specified events occurs, the flag bit is set. If the corresponding enable bit is 0, the interrupt request is blocked. If the enable bit is 1, the interrupt request is forwarded to the CPU as an I2C interrupt request. As an alternative, the CPU can poll all of the bits shown in [Table 26-2](#).

**Table 26-2. Interrupt Requests Generated by I2C Module**

Flag	Name	Generated
AL	Arbitration-lost interrupt	Generated when the I2C module has lost an arbitration contest with another master-transmitter
NACK	No-acknowledge interrupt	Generated when the master I2C does not receive an acknowledge from the receiver
ARDY	Register-access-ready interrupt	Generated when the previously programmed address, data and command have been performed and the status bits have been updated. The interrupt is used to notify the device that the I2C registers are ready to be accessed.
RXRDY	Receive-data-ready interrupt	Generated when the received data in the receive-shift register (I2CSR) has been copied into the data receive register (I2CDRR). The RXRDY bit can also be polled by the device to determine when to read the received data in the I2CDRR.
TXRDY	Transmit-data-ready interrupt	Generated when the transmitted data has been copied from the data transmit register (I2CDXR) into the transmit-shift register (I2CXSR). The TXRDY bit can also be polled by the device to determine when to write the next data into I2CDXR.
SCD	Stop-condition-detect interrupt	Generated when a STOP condition has been detected.
AAS	Address-as-slave interrupt	Generated when the I2C has recognized its own slave address or an address of all zeroes.

The interrupt vector register (I2CIVR) contains the binary-coded-interrupt vector that indicates the highest priority interrupt that is pending and enabled. When I2CIVR is read, the corresponding interrupt flags for AL, NACK and SCD are automatically cleared, if their interrupts are enabled. Reading the I2CIVR will not clear the AAS, ARDY, RXRDY, or TXRDY interrupt pending flags. Please see section [Section 26.6.3, I2C Status Register \(I2CSTR\)](#) for the method to clear these four flags.

If more than one interrupt is pending, a new interrupt will be generated for the next highest priority pending interrupt when you re-enable the I2C interrupt.

A transmit interrupt is generated just after the START condition in master transmitter mode. This ensures that the CPU will get an interrupt even if no slave returns an ACK to the slave address following the START condition.

It is important to note that when the I2C is configured to generate interrupts as a slave transmitter and the backward compatibility mode (BCM) bit is set to 1, an extra transmit interrupt occurs. The application should monitor the ACK from the master to determine whether to load another byte into I2CDXR. See Appendix A: Backward Compatibility Mode, for more information.



### 26.5.2 DMA Controller Events

The I2C module has two events that use the DMA controller to synchronously read received data (I2CREVNT) from I2CDRR, and synchronously write data (I2CWEVNT) to the transmit buffer, I2CDXR. The read and write events have the same timing as I2CRRDY (I2CRINT) and I2CXRDY (I2CXINT), respectively.

The CPU or the DMA controller reads the received data from I2CDRR and writes the data to be transmitted to I2CDXR. The RXRDY bit is automatically cleared when the DMA controller reads the I2CDRR register, and the TXRDY bit is automatically cleared when the DMA controller writes to the I2CDXR register.

Data written to I2CDXR is copied to I2CXSR and shifted out from the SDA pin when the I2C module is configured as a transmitter. When the I2C module is configured as a receiver, received data is shifted into ICRSR and copied to I2CDRR, which can be read by the CPU or the DMA controller.

A Transmit event (I2CWEVNT) is generated after a START condition in master transmitter mode. This ensures that the DMA gets an event even if no slave returns an ACK to the slave address following the START condition.

---

**NOTE: Unexpected DMA transmit and receive event**

An unexpected DMA transmit event (ICXEVT) and a DMA receive event (ICXRDY) are generated in 10-bit, master transmit, repeat mode. This event occurs soon after the start condition but before the first bit of the address is transmitted. In this event, no DMA activity should be initiated without the slave ACK being received.

---

### 26.5.3 I2C Enable/Disable

The I2C module can be enabled or disabled with the I2C reset enable bit (IRS) in the I2C module register (I2CMDR). This occurs in one of two ways:

- Write 0 to the I2C reset bit (IRS) in I2CMDR. All status bits are forced to the default values and the I2C mode remains disabled until IRS is changed to 1. The SDA and SCL pins are in the high impedance state.
- Initiate a device reset by driving the  $\overline{\text{PORRST}}$  pin low. The entire device is reset and is held in the reset state until the pin is released and is driven high. When PORRST is released, all I2C module registers are reset to their default values. The IRS bit is forced to 0, which resets the I2C module. The I2C module stays in the reset state until a 1 is written to the IRS bit.

IRS must be 0 while the I2C module is being configured. Forcing IRS to 0 can be used to save power and also clear error conditions.

### 26.5.4 General Purpose I/O

Both of the I2C pins can be programmed to be general-purpose I/O pins via the I2C pin control registers (I2CPFNC, I2CDIR, I2CDOOUT, and I2CDIN).

When the I2C module is not used, the I2C pins may be programmed to be either general purpose input or general-purpose output pins. This function is controlled in the I2CDIR and I2CPFNC registers. Note that each pin can be programmed to be either an I2C pin or a GIO pin.

If the I2C function is to be used, the application software must ensure that each pin is configured as an I2C pin and not a GIO pin, or else unexpected behavior may result.

### 26.5.5 Pull Up/Pull Down Function

I2C module pins can have either an active pull up or active pull down that makes it possible to leave the pins unconnected externally. The pins can be programmed to have the active pull function enabled or disabled by writing to the corresponding bit in the I2CPDIS register. Please see the device-specific data sheet for the default internal pull (pull-up, pull-down or no pull) on the pins.

The pull on the pins is programmable to a setting other than the default internal pull as specified in the data sheet. The pins can be programmed to have either an active pull up or an active pull down function by writing to the corresponding bit in I2CPSEL register. The pull up/pull down function is active on the pin only when the pull enabled is programmed in the I2CPDIS register.

The pull up/pull down functions are deactivated when a bidirectional pin is configured as an output. At system reset, the pull up function of all the pins is disabled. Please see the device-specific data sheet for the current supplied by the pull up/pull down.

### **26.5.6 Open Drain Function**

The I2C pins can be programmed to include an open drain function when they are configured as output pins. This is done by writing to the corresponding bit of the I2CPDR register. When the open drain function is enabled, a low value (0) written to the data output register forces the pin to a low output voltage ( $V_{OL}$  or lower), whereas a high value (1) written to the data output register forces the pin to a high impedance state. The open drain function is disabled when the pin is configured as an input pin.

## 26.6 I2C Registers

Table 26-3 provides a summary of the registers.

The upper word (upper 16 bits) of the below registers are all read as 0s. Writes have no effect on these bits. For a more detailed description of the individual bits, see the following sections.

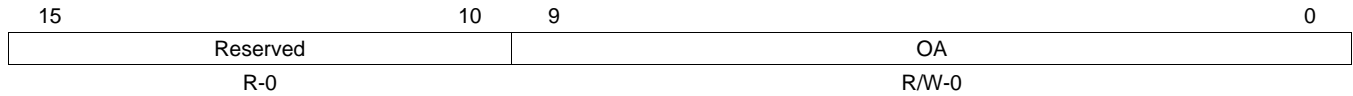
**Table 26-3. I2C Control Register Summary**

Offset	Acronym	Register Description	Section
00	I2COAR	I2C Own Address Manager	<a href="#">Section 26.6.1</a>
04h	I2CIMR	I2C Interrupt Mask Register	<a href="#">Section 26.6.2</a>
08h	I2CSTR	I2C Status Register	<a href="#">Section 26.6.3</a>
0Ch	I2CCKL	I2C Clock Divider Low Register	<a href="#">Section 26.6.4</a>
10h	I2CCKH	I2C Clock Control High Register	<a href="#">Section 26.6.5</a>
14h	I2CCNT	I2C Data Count Register	<a href="#">Section 26.6.6</a>
18h	I2CDRR	I2C Data Receive Register	<a href="#">Section 26.6.7</a>
1Ch	I2CSAR	I2C Slave Address Register	<a href="#">Section 26.6.8</a>
20h	I2CDXR	I2C Data Transmit Register	<a href="#">Section 26.6.9</a>
24h	I2CMRDR	I2C Mode Register	<a href="#">Section 26.6.10</a>
28h	I2CIVR	I2C Interrupt Vector Register	<a href="#">Section 26.6.11</a>
2Ch	I2CEMDR	I2C Extended Mode Register	<a href="#">Section 26.6.12</a>
30h	I2CPSC	I2C Prescale Register	<a href="#">Section 26.6.13</a>
34h	I2CPID1	I2C Peripheral ID Register 1	<a href="#">Section 26.6.14</a>
38h	I2CPID2	I2C Peripheral ID Register 2	<a href="#">Section 26.6.15</a>
3Ch	I2CDMACR	I2C DMA Control Register	<a href="#">Section 26.6.16</a>
48h	I2CPFNC	I2C Pin Function Register	<a href="#">Section 26.6.17</a>
4Ch	I2CPDIR	I2C Pin Direction Register	<a href="#">Section 26.6.18</a>
50h	I2CDIN	I2C Data Input Register	<a href="#">Section 26.6.19</a>
54h	I2CDOUT	I2C Data Output Register	<a href="#">Section 26.6.20</a>
58h	I2CDSET	I2C Data Set Register	<a href="#">Section 26.6.21</a>
5Ch	I2CDCLR	I2C Data Clear Register	<a href="#">Section 26.6.22</a>
60h	I2CPDR	I2C Pin Open Drain Register	<a href="#">Section 26.6.23</a>
64h	I2CPDIS	I2C Pull Disable Register	<a href="#">Section 26.6.24</a>
68h	I2CPSEL	I2C Pull Select Register	<a href="#">Section 26.6.25</a>
6Ch	I2CSRS	I2C Pins Slew Rate Select Register	<a href="#">Section 26.6.26</a>

### 26.6.1 I2C Own Address Manager (I2COAR)

The 16-bit memory-mapped I2C own address register is used to specify its own address. [Figure 26-13](#) and [Table 26-4](#) describe this register.

**Figure 26-13. I2C Own Address Manager Register (I2COAR) [Offset = 00]**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 26-4. I2C Own Address Manager Register (I2COAR) Field Descriptions**

Bit	Field	Value	Description
15-10	Reserved	0	Read returns 0. Writes have no effect.
9-0	OA	0-3FFh	Own address.  These bits reflect the bus address of the I2C module. When the expand address (XA) bit I2CMDR.8 is set to 1, the I2C is in expand address mode (10-bit addressing mode). In either 7 or 10-bit address mode, all 10-bits are both readable and writable. Bits 7, 8, and 9 should only be used in 10-bit address mode. <a href="#">Table 26-5</a> provides the correct modes for these bits. Note that the user can program the I2C own address to any value as long as it does not conflict with other components in the system.

**Table 26-5. Correct Mode for Bits OA.9:0**

Bits Used	Mode	Value of XA
OA.6:0	7 Bit Addressing	0
OA.9:0	10 Bit Addressing	1

### 26.6.2 I2C Interrupt Mask Register (I2CIMR)

The 7-bit memory mapped I2C interrupt mask register is used by the device to enable/disable the interrupts. [Figure 26-14](#) and [Table 26-6](#) describe this register.

**Figure 26-14. I2C Interrupt Mask Register (I2CIMR) [Offset = 04h]**

Reserved							
R-0							
7	6	5	4	3	2	1	0
Reserved	AASEN	SCDEN	TXRDYEN	RXRDYRN	ARDYEN	NACKEN	ALEN
R-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 26-6. I2C Interrupt Mask Register (I2CIMR) Field Descriptions**

Bit	Field	Value	Description
15-7	Reserved	0	Read returns 0. Writes have no effect.
6	AASEN	0	The AASEN interrupt is disabled.
		1	The AASEN interrupt is enabled.
5	SCDEN	0	The SCDEN interrupt is disabled.
		1	The SCDEN interrupt is enabled.
4	TXRDYEN	0	The TXRDYEN interrupt is disabled.
		1	The TXRDYEN interrupt is enabled.
3	RXRDYEN	0	The RXRDYEN interrupt is disabled.
		1	The RXRDYEN interrupt is enabled.
2	ARDYEN	0	The ARDYEN interrupt is disabled.
		1	The ARDYEN interrupt is enabled.
1	NACKEN	0	The NACKEN interrupt is disabled.
		1	The NACKEN interrupt is enabled.
0	ALEN	0	The ALEN interrupt is disabled.
		1	The ALEN interrupt is enabled.

### 26.6.3 I2C Status Register (I2CSTR)

Figure 26-15 and Table 26-7 describe this register.

**Figure 26-15. I2C Status Register (I2CSR) [Offset = 08h]**

15	14	13	12	11	10	9	8
Reserved	SDIR	NACKSNT	BB	RSFULL	XSMT	AAS	AD0
R-0	R/W-0	R/W-0	R-0	R-0	R/W-1	R-0	R-0
7	6	5	4	3	2	1	0
Reserved		SCD	TXRDY	RXRDY	ARDY	NACK	AL
R-0		R/W-0	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 26-7. I2C Status Register (I2CSTR) Field Descriptions**

Bit	Field	Value	Description
15	Reserved	0	Read returns 0. Writes have no effect.
14	SDIR		Slave direction: Setting this bit to 1 indicates that the I2C slave is a transmitter. Clearing this bit to 0 indicates that the I2C is a master transmitter/receiver or a slave receiver. This bit is also cleared by the STOP or START conditions. In DLB mode (in which the configuration should be master-transmitter slave-receiver), this bit is cleared to 0. <b>Writing a 1 to this bit will clear it.</b>
		0	The I2C is a master transmitter/receiver or a slave receiver.
		1	The I2C is a slave transmitter.
13	NACKSNT		No acknowledge sent: This bit is set to 1 to indicate that a no acknowledgement (NACK) has been sent because the NACKMOD bit was set to 1. <b>Writing a 1 to this bit will clear it.</b>
		0	A NACK has not been sent.
		1	A NACK was sent because the NACKMOD was set to 1.
12	BB		Bus busy: This bit indicates the state of the serial bus. On reception of a START condition or if the I2C detects a low state on I2CSCL, the device sets BB = 1. If the nIRS is set to 1 during transaction between other I2C devices, the BB bit is set at the first falling edge of SCL or START condition. BB is cleared to 0 after the reception of a STOP condition. BB is kept to 0 regardless of the SCL state when the I2C is in reset (nIRS = 0).
		0	The bus is free.
		1	The bus is busy.
11	RSFULL		Receiver shift full: This bit is set to 1 to indicate that the receiver has experienced overrun. Overrun occurs when the receive shift register is full and I2CDRR has not been read since the receive shift register to I2CDRR transfer. The contents of I2CDRR are not lost. The I2C core logic is holding for I2CDRR read access. This bit is also set when, in master-repeat-mode, the I2C receives a byte of data. There is no difference between RXRDY and RSFULL in this case. The I2C master will not continue the transfer as long as the received data is in the I2CDRR or receive shift register. <b>RSFULL is cleared when reading the I2CDRR, resetting the I2C (IRS_=0), or resetting the device.</b>
		0	No overrun has occurred.
		1	An overrun has occurred.

**Table 26-7. I2C Status Register (I2CSTR) Field Descriptions (continued)**

Bit	Field	Value	Description
10	XSMT		Transmit shift empty not: This bit is cleared to 0 to indicate that the transmitter has experienced underflow. Underflow occurs when the transmit shift register is empty and I2CDXR has not been loaded since the last I2CDXR to transmit shift register transfer. The I2C core logic is waiting for I2CDXR write access. <b>XMSTn is set to 1 as a result of writing to I2CDXR, by resetting the I2C block(IRS=0), or by resetting the device.</b> In repeat mode, if the I2C in master transmitter mode is holding transfer with XSMTn = 0 (i.e. waiting for further action) and the STT or STP bit is set, XSMTn is set to 1 by hardware.
		0	An underflow has occurred.
9	AAS	1	No underflow has occurred.
		0	Address as slave <b>This bit cannot be cleared by writing a 1 to the bit or by reading the I2CIVR register.</b> This bit is cleared by a STOP condition or detection of any address byte that does not match I2COAR.
8	AD0	1	This bit is set to 1 by the device when it has recognized its own slave address or an address of all zeros (general call).
		0	Address zero status A START or STOP condition was detected. No general call was detected.
7-6	Reserved	0	An address of all zeros (general call) was detected. Read returns 0. Writes have no effect.
5	SCD		Stop condition detect interrupt flag: This bit is set to 1 when the I2C receives or sends a STOP condition.
		0	<b>This bit is cleared to 0 by writing a 1 to this bit or reading the value 0x0006 from I2CIVR. Writing a 1 to this bit will clear the value 0x0006 from I2CIVR.</b> No STOP condition has been sent or received.
4	TXRDY	1	A STOP condition has been sent or received.
		0	Transmit data ready interrupt flag: This bit is set to 1 to indicate when data in the transmit data register, I2CDXR, has been copied into the transmit shift register. This bit can also be polled by the device to indicate when to write the next transmitted data into the I2CDXR. <b>Writing a 1 to this bit will set it.</b> <b>This bit is cleared to 0 and code 0x0005 in I2CIVR is cleared when the I2CDXR is written. This bit cannot be cleared by reading the I2CIVR register.</b>
3	RXRDY	1	I2CDXR contains data to transmit.
		0	I2CDXR is empty.
		0	Receive data ready interrupt flag: This bit is set to 1 to indicate when the data in the receive shift register has been copied into the data receive register (I2CDRR). This bit can also be polled by the device to indicate when to read the received data in the I2CDRR.
		1	<b>Writing a 1 to this bit or reading from I2CDRR will clear this bit, and will also clear code 0x0004 from I2CIVR. This bit cannot be cleared by reading the I2CIVR register.</b> The I2CDRR has been read.
		1	The received data has been written into the I2CDRR.

**Table 26-7. I2C Status Register (I2CSTR) Field Descriptions (continued)**

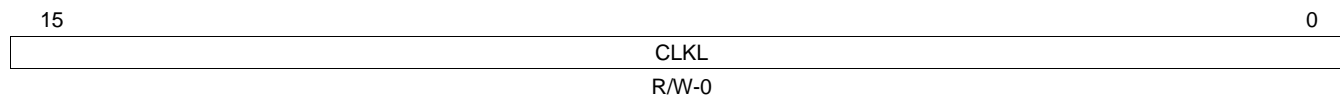
Bit	Field	Value	Description
2	ARDY	0 1	<p>Register access ready interrupt flag:</p> <p>This bit is set to 1 when the previously programmed address, data and command has been performed and the status bit has been updated. The flag is used by the device to indicate that the I2C registers are ready to be accessed again.</p> <p><b>This bit is automatically cleared by hardware when writing data to I2CDXR in transmit mode, reading data from I2CDRR in receive mode, or setting STT or STP bit. Writing a 1 to this bit will clear this bit. This bit cannot be cleared by reading the I2CIVR register.</b></p> <p>When RM=0, ARDY is set when I2CCNT is passed 0 if STP register bit has not been set. When RM=1, ARDY is set at each byte end.</p> <p>When FDF is 0, ARDY is asserted after the ACK for the slave address. When FDF is 1, there is no slave address. Therefore, ARDY is asserted after sending the start condition.</p> <p><i>Nonrepeat mode, (RM = 0):</i> I2C registers are not ready to be accessed.  <i>Repeat mode (RM = 1):</i> I2C registers are not ready to be accessed.</p> <p><i>Nonrepeat mode, (RM = 0):</i> ICCNT passes 0 (if STP bit has not been set).  <i>Repeat mode (RM = 1):</i> The end of each byte was transmitted from I2CDXR.</p>
1	NACK	0 1	<p>No acknowledgement interrupt:</p> <p>This bit is set to 1 when the master I2C does not receive an acknowledgement from the receiver. This bit is set only when the I2C has received a no-acknowledge in master mode. This bit is NOT set by no-acknowledgement after Start byte. In master start byte mode, the first byte (address of all zeroes) receives a NACK but does not clear the stop bit.</p> <p><b>Writing a 1 to this bit or reading the value 0x0002 from I2CIVR will clear this bit.</b></p> <p>0 An acknowledge was detected.</p> <p>1 No acknowledge was detected or the I2C is operating in the general call, even though an acknowledgement was received. This value clears the STP bit.</p>
0	AL	0 1	<p>Arbitration lost interrupt flag:</p> <p>This bit is set to 1 when arbitration has been lost.</p> <p><b>Writing a 1 to this bit or reading the value 0x0001 from I2CIVR will clear this bit.</b></p> <p>0 No loss of arbitration has been detected.</p> <p>1 The device in the master transmitter mode senses it has lost an arbitration. This occurs when two or more transmitters start a transmission almost simultaneously or when the I2C attempts to start a transfer while BB=1. When this is set to 1 due to arbitration lost, the device becomes a slave receiver and the MST, STT and STP bits in I2CMDR are cleared to 0.</p>



### 26.6.4 I2C Clock Divider Low Register (I2CCKL)

The I2C clock divider low register is a 16-bit memory mapped register used to divide the master clock down to obtain the I2C serial clock low time. [Figure 26-16](#) and [Table 26-8](#) describe this register.

**Figure 26-16. I2C Clock Divider Low Register (I2CCKL) [Offset = 0Ch]**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

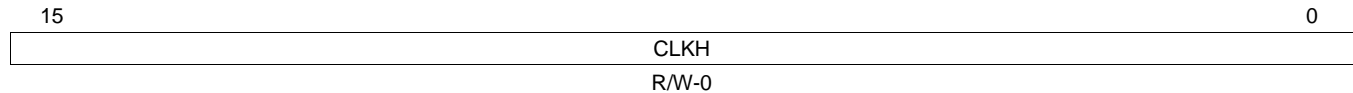
**Table 26-8. I2C Clock Divider Low Register (I2CCKL) Field Descriptions**

Bit	Field	Value	Description
15-0	CLKL	0-FFFFh	<p>Low time clock division factor: Used to divide down the module clock to create the low time portion of the master clock signal that will appear on the SCL pin.:</p> $LowTime = \left( \frac{I2CCKL + d}{ModuleClockFrequency} \right) \quad (61)$ <p>where <i>d</i> is the value that depends on the I2CPSC (see <a href="#">Section 26.1.3</a>). This register must be configured while the I2C is still in reset (nIRS = 0).</p>

### 26.6.5 I2C Clock Control High Register (I2CCKH)

The I2C clock divider high register is a 16-bit memory mapped register used to divide the master clock down to obtain the I2C serial clock high time. [Figure 26-17](#) and [Table 26-9](#) describe this register.

**Figure 26-17. I2C Clock Control High Register (I2CCKH) [Offset = 10h]**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

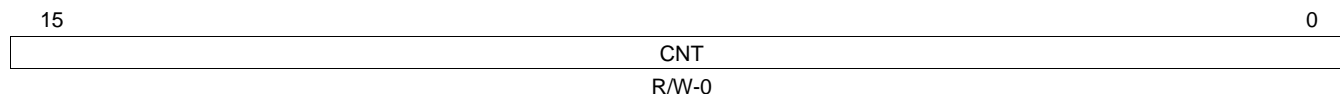
**Table 26-9. I2C Clock Control High Register (I2CCKH) Field Descriptions**

Bit	Field	Value	Description
15-0	CLKH	0-FFFFh	<p>High time clock division factor: Used to divide down the module clock to create the high time portion of the master clock signal that will appear on the SCL pin:</p> $HighTime = \left( \frac{I2CCLKH + d}{ModuleClockFrequency} \right) \quad (62)$ <p>where <i>d</i> is the value that depends on the I2CPSC (see <a href="#">Section 26.1.3</a>). This register must be configured while the I2C is still in reset (nIRS = 0).</p>

### 26.6.6 I2C Data Count Register (I2CCNT)

The I2C data count register is a 16-bit memory mapped register used to count received or transmitted data bytes. This register is also used to generate the STOP condition which terminates the transfer after the counter reaches zero. [Figure 26-18](#) and [Table 26-10](#) describe this register.

**Figure 26-18. I2C Data Count Register (I2CCNT) [Offset = 14h]**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

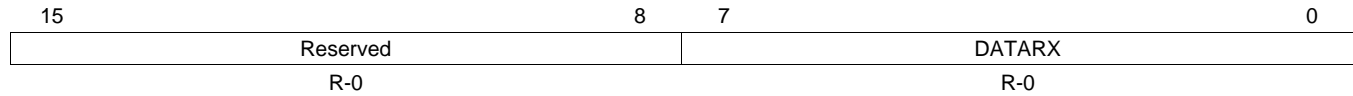
**Table 26-10. I2C Data Count Register (I2CCNT) Field Descriptions**

Bit	Field	Value	Description
15-0	CNT	0-FFFFh	Data counter: This down counter is used to generate a stop condition if a stop condition is specified (STP=1). <b>Note: ICCNT is a don't care when RM is set to 1.</b>
		00	The data counter is 65536.
		1h	The data counter is 1.

### 26.6.7 I2C Data Receive Register (I2CDRR)

The I2C data receive register is a 16-bit memory mapped register used by the device to read the received data. [Figure 26-19](#) and [Table 26-11](#) describe this register.

**Figure 26-19. I2C Data Receive Register (I2CDRR) [Offset = 18h]**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

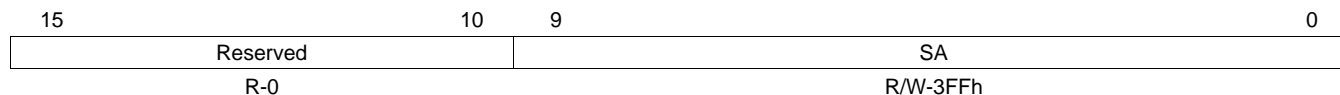
**Table 26-11. I2C Data Receive Register (I2CDRR) Field Descriptions**

Bit	Field	Value	Description
15-8	Reserved	0	Read returns 0. Writes have no effect.
7-0	DATARX	0-FFh	Receive data: A read from this register clears the RXRDY bit and clears code 4h from the I2CIVR register.

### 26.6.8 I2C Slave Address Register (I2CSAR)

The I2C slave address register is a 16-bit memory-mapped register used to specify the address of the slave device to communicate to on the I2C bus. [Figure 26-20](#) and [Table 26-12](#) describe this register.

**Figure 26-20. I2C Slave Address Register (I2CSAR) [Offset = 1Ch]**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 26-12. I2C Slave Address Register (I2CSAR) Field Descriptions**

Bit	Field	Value	Description
15-10	Reserved	0	Read returns 0. Writes have no effect.
9-0	SA		7- or 10-bit programmable slave address: In either mode, all 10-bits are readable and writable. Bits 7, 8, and 9 should only be used in 10-bit address mode. <a href="#">Table 26-13</a> illustrates the correct mode for each bit.

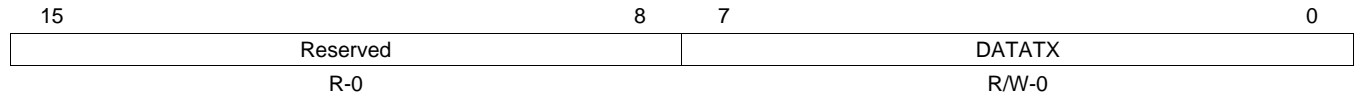
**Table 26-13. Correct Modes for Bits SA(9–0)**

Bits Used	Mode	Value of XA
SA(6–0)	7-bit addressing	0
SA(9–0)	10-bit addressing	1

### 26.6.9 I2C Data Transmit Register (I2CDXR)

Figure 26-21 and Table 26-14 describe this register.

**Figure 26-21. I2C Data Transmit Register (I2CDXR) [Offset = 20h]**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 26-14. I2C Data Transmit Register (I2CDXR) Field Descriptions**

Bit	Field	Value	Description
15-8	Reserved	0	Read returns 0. Writes have no effect.
7-0	DATATX	0-FFh	Transmit data: Data written to this register will be transmitted on the I2C bus. A write to this register clears the TXRDY bit and clears code 0x05 from the I2CIVR register.

## 26.6.10 I2C Mode Register (I2CMDR)

Figure 26-22 and Table 26-15 describe this register.

**Figure 26-22. I2C Mode Register (I2CMDR) [Offset = 24h]**

15	14	13	12	11	10	9	8
NACKMOD	FREE	STT	Reserved	STP	MST	TRX	XA
R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	0	
RM	DLB	nIRS	STB	FDF	BC		
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0		

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 26-15. I2C Mode Register (I2CMDR) Field Descriptions**

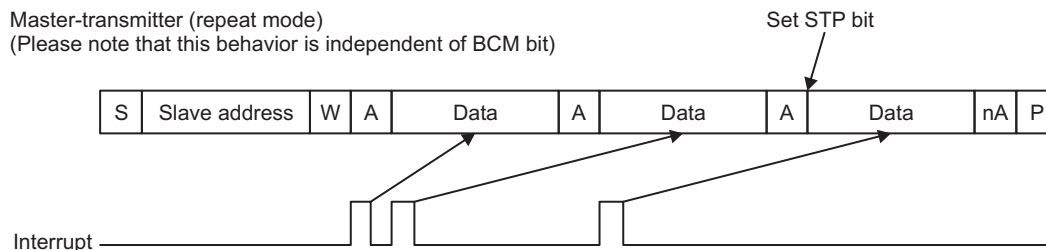
Bit	Field	Value	Description
15	NACKMOD	0 1	<p>No-acknowledge (NACK) mode:</p> <p>This bit is used to send an acknowledge (ACK) or a no-acknowledge (NACK) to the transmitter. This bit is only applicable when the I2C is in receiver mode. In master receiver mode, when the internal data counter decrements to zero, the I2C sends a NACK. The master receiver I2C finishes a transfer when it sends a NACK. The I2C ignores ICCNT when NACKMOD is 1. The NACKMOD bit should be set before the rising edge of the last data bit if a NACK must be sent, and this bit is cleared once a NACK has been sent.</p> <p>0 The I2C sends an ACK to the transmitter during the acknowledge cycle.</p> <p>1 The I2C sends a NACK to the transmitter during the acknowledge cycle.</p>
14	FREE	0 1	<p>Free running bit:</p> <p>This bit is used to determine the state of the I2C when a breakpoint is encountered in the high level language (HLL) debugger.</p> <p>0 The I2C stops immediately if SCL is low and keeps driving SCL low if the I2C master is a transmitter/receiver. If SCL is high, I2C waits until SCL becomes low and then stops. If the I2C is a slave, it will stop when the transmission/reception completes.</p> <p>1 The I2C runs free.</p>
13	STT	0 1	<p>Start condition:</p> <p>The start condition bit works with the STP bit (master only mode). The STT and STP bits are configured to generate different transfer formats (see Table 26-16). The STT and STP bits can be used to terminate the repeat mode. This bit takes one I2C module clock cycle to set.</p> <p>0 The STT is reset to 0 by the hardware after the START condition has been generated.</p> <p>1 STT is set to 1 by the device to generate a START condition. In master mode, setting STT to 1 generates a START condition.</p>
12	Reserved	0	Read returns 0. Writes have no effect.
11	STP	0 1	<p>Stop condition:</p> <p>1(Master mode only) This bit can be set to a 1 by the CPU to generate a stop condition. It is reset to 0 by the hardware after the stop condition has been generated. The stop condition is generated when ICCNT passes 0 when the I2C is in non-repeat mode (RM=0). In repeat mode (RM=1), the stop condition is generated if STP bit is 1. In transmitter mode, I2CTXRDY needs to be 1 (i.e. you have to set STP bit unless you write data into I2CDXR).</p> <p>0 STP is reset to 0 by the hardware after the STOP condition has been generated.</p> <p>1 STP is set to 1 by the device to generate a STOP condition.</p>
10	MST	0 1	<p>Master/slave mode bit:</p> <p>This bit determines whether the module will operate in master or slave mode; see Table 26-17. This bit is cleared after generating a STOP condition. The BB bit is cleared first, and MST bit is cleared second. Before starting the next transaction in master mode, this bit must be confirmed to be cleared.</p> <p>0 The module is in the slave mode and the clock is received from the master device.</p> <p>1 The module is in the master mode and it generates the clock. This bit is cleared when the transfer has completed.</p>

**Table 26-15. I2C Mode Register (I2CMDR) Field Descriptions (continued)**

Bit	Field	Value	Description
9	TRX	0 1	<p>Transmit/receive bit:</p> <p>This bit determines the direction of data transmission of the I2C module. See <a href="#">Table 26-17</a>.</p> <p>0 The module is in the receive mode and data on the SDA line is shifted into the data register I2CDRR.</p> <p>1 The module is in the transmit mode and the data in the I2CDXR is shifted out on the SDA line.</p>
8	XA	0 1	<p>Expand address enable bit:</p> <p>This bit controls the addressing mode. When XA is set to 1, the I2C does not support the combined format in master mode operations. However, the I2C will acknowledge and support the formats when configured as a slave. This bit needs to be configured even if the I2C is in slave mode.</p> <p>0 The mode is set to 7-bit addressing mode (normal address mode).</p> <p>1 The mode is set to 10-bit addressing mode (expanded address mode).</p>
7	RM	0 1	<p>Repeat mode enable bit: (Master mode only)</p> <p>This bit is a 'don't care' if the module is configured in slave mode (MST = 0); see <a href="#">Table 26-16</a>. Each time a byte of data is received, the user should decide whether or not to continue receiving more data. See <a href="#">Figure 26-23</a> for a diagram of this function.</p> <p>0 The mode is not in repeat mode.</p> <p>1 In repeat mode, data is continuously transmitted out of the ICDXR or received into the ICDRR until the STP bit is set to 1 regardless of ICCNT value. See <a href="#">Table 26-16</a> for module conditions.</p>
6	DLB	0 1	<p>Digital loop back enable bit:</p> <p>This bit disables or enables the digital loopback mode of the I2C. This bit only applies in Master transmitter mode.</p> <p>0 Digital loop back mode is disabled.</p> <p>1 Digital loop back mode is enabled. In digital loop back mode, data transmitted out of the I2CDXR will be received in the I2CDRR. The address of the I2COAR is output on SDA.</p>
5	nIRS	0 1	<p>I2C reset enable bit:</p> <p>When set to 0, this bit will place all status registers in this module to their default state. Resetting nIRS during a data transfer can hang the I2C bus.</p> <p>0 I2C is in reset.</p> <p>1 I2C is out of reset.</p>
4	STB	0 1	<p>Start byte mode enable bit: (Master mode only)</p> <p>The Start byte mode bit is set to 1 by the CPU to configure the I2C in Start byte mode. The I2C sends "00000001" regardless of the I2CSAR value. Refer to the Philips I2C specification for more details.</p> <p>0 The module is not in START byte mode.</p> <p>1 The module is in START byte mode.</p>
3	FDf	0 1	<p>Free data format enable bit:</p> <p>This bit configures the module to operate in free data format mode (see <a href="#">Table 26-17</a>) in both master and slave modes. When FDF is 0, ARDY is asserted after ACK for the slave address. When FDF is 1, there is no slave address. Therefore, ARDY is asserted after sending the start condition. FDF mode is not supported in digital loop back mode.</p> <p>0 The module is not in free data format mode.</p> <p>1 The module is in free data format mode.</p>
2-0	BC		<p>Bit count:</p> <p>This bit defines the number of bits starting from the LSB (excluding the acknowledge bit) that are sent on the bus when data is written to the data transmit register.</p> <p>If the bits BC0, BC1, and BC2 are all 0, then the number of bits sent on the bus is 8. If the bit count bits are a non-zero value, then the number of bits sent on the bus is that value. The value "001" is reserved. When performing a transfer using the bit count of, for example, n (where n is nonzero), only the n least significant bits in the data receive register are valid and correct. The rest of the bits should be disregarded. See <a href="#">Table 26-18</a> for more information.</p>



**Figure 26-23. Typical Timing Diagram of Repeat Mode**



**Table 26-16. I2C Module Condition, Bus Activity, and Mode**

RM	STT	STP	Condition	Bus Activities <sup>(1)</sup>	Mode
0	0	0	Idle	None	N/A
0	0	1	Stop	P	N/A
0	1	0	(Repeat) Start	S-A-D..(n)..D	Repeat n
0	1	1	(Repeat) Start-Stop	S-A-D..(n)..D-P	Repeat n
1	0	0	Idle	none	N/A
1	0	1	Stop	P	N/A
1	1	0	(Repeat) Start	S-A-D-D-D-....	Continuous
1	1	1	Reserved	none	N/A

<sup>(1)</sup> P = Stop condition; S = Start condition; A = Acknowledge bit; D = data

**Table 26-17. I2C Module Operating Modes**

FDF	MST	TRX	Operating Mode
0	0	x	Slave in non-FDF mode
0	1	0	Master receive in non-FDF mode
0	1	1	Master transmit in non-FDF mode
1	0	0	Slave receive in FDF mode
1	0	1	Slave transmit in FDF mode
1	1	0	Master receive in FDF mode
1	1	1	Master transmit in FDF mode

**Table 26-18. Number of Bits Sent on Bus**

BC2	BC1	BC0	Bits in FDF	Bits with ACK
0	0	0	8	9
0	0	1	NA (reserved)	NA(reserved)
0	1	0	2	3
0	1	1	3	4
1	0	0	4	5
1	0	1	5	6
1	1	0	6	7
1	1	1	7	8

### 26.6.11 I2C Interrupt Vector Register (I2CIVR)

The I2C interrupt vector register is a 16-bit memory-mapped register used to indicate the occurrence of an interrupt. [Figure 26-24](#) and [Table 26-19](#) describe this register.

**Figure 26-24. I2C Interrupt Vector Register (I2CIVR) [Offset = 28h]**

15	12	11	8	7	3	2	0
Reserved		TESTMD		Reserved		INTCODE	
R-0		R/W-0		R-0		R/WC-0	

LEGEND: R/W = Read/Write; R = Read only; C = Clear; -n = value after reset

**Table 26-19. I2C Interrupt Vector Register (I2CIVR) Field Descriptions**

Bit	Field	Value	Description
15-12	Reserved	0	Read returns 0. Writes have no effect.
11-8	TESTMD	0-3h	Reserved for internal testing.
7-3	Reserved	0	Read returns 0. Writes have no effect.
2-0	INTCODE	0-3h	<p>Interrupt Code Bits:</p> <p>This binary coded interrupt vector indicates which interrupt has occurred. If there is more than one interrupt pending, reading I2CIVR provides the vector for the highest priority interrupt that is pending.</p> <p>Reading the I2CIVR will clear the corresponding flags in I2CSTR for AL, NACK and SCD as long as those interrupts are enabled. A new interrupt will be generated for each pending source.</p> <p>Reading I2CIVR will clear the INTCODE for AL, NACK, SCD, AAS, RXRDY and TXRDY. Reading I2CIVR <b>will not</b> clear the INTCODE for ARDY.</p> <p>The INTCODE for certain codes can also be cleared by either writing a 1 to the corresponding interrupt flag bits in I2CSTR, or by reading and writing to the receive or transmit registers. Please see <a href="#">Section 26.6.3, I2C Status Register (I2CSTR)</a> for more details.</p> <p>Users must read (clear) the I2CIVR before doing another start otherwise the I2CIVR could contain incorrect (old interrupt flag) value.</p>

**Table 26-20. Interrupt Codes for INTCODE Bit**

Code	INTCODE(2-0)	Interrupt Occurred
00	000	none
01h	001 (highest priority)	Arbitration lost (AL)
02h	010	No acknowledgement (NACK)
03h	011	Receive access ready (ARDY)
04h	100	Receive data ready (RXRDY)
05h	101	Transmit data ready (TXRDY)
06h	110	Stop condition detection (SCD)
07h	111 (lowest priority)	Address as slave (AAS)

### 26.6.12 I2C Extended Mode Register (I2CEMDR)

The I2C extended mode register is a 16-bit memory-mapped register that contains additional mode select bits. [Figure 26-25](#) and [Table 26-21](#) describe this register.

**Figure 26-25. I2C Extended Mode Register (I2CEMDR) [Offset = 2Ch]**

15	Reserved	2	1	0
	R-0		IG NACK	BCM
			R/W-0	R/W-1

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

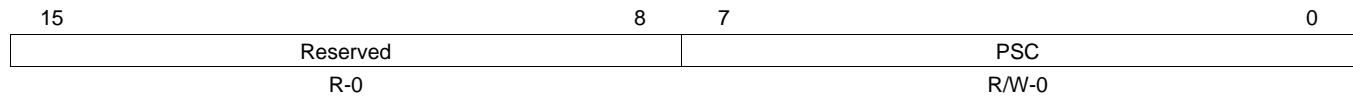
**Table 26-21. I2C Extended Mode Register (I2CEMDR) Field Descriptions**

Bit	Field	Value	Description
15-2	Reserved	0	Read returns 0. Writes have no effect.
1	IGNACK	0	The master transmitter will operate normally, discontinue the data transfer, and set the ARDY and NACK status bits when a NACK signal is received from the slave.
		1	The master transmitter will ignore a NACK received from the slave.
0	BCM		Backwards compatibility mode: When set to 1, the I2C is compatible with previous versions of the I2C. This means the TXRDY interrupt is generated in slave-transmit mode when TXRDY is set and the I2C needs more data to transmit. This behavior causes an extra TXRDY interrupt to be generated because the I2C recognizes the end of transfer after generating an interrupt for the next byte of data.
		0	When BCM is 0, the TXRDY interrupt in slave-transmit mode is generated when XSMT = 1. In this case, the I2C generates an interrupt for the next byte after receiving the ACK from previous data. The setting of this bit only applies to slave transmit mode.
		0	The I2C is <b>not</b> in compatibility mode.
		1	The I2C is in compatibility mode.

### 26.6.13 I2C Prescale Register (I2CPSC)

The I2C prescaler register is a 16-bit memory-mapped register used for dividing down the VBUS\_CLK to obtain a module clock frequency between 6.7 MHz and 13.3 MHz. [Figure 26-26](#) and [Table 26-22](#) describe this register.

**Figure 26-26. I2C Prescale Register (I2CPSC) [Offset = 30h]**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

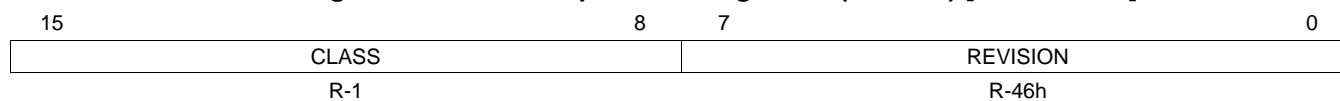
**Table 26-22. I2C Prescale Register (I2CPSC) Field Descriptions**

Bit	Field	Value	Description
15-8	Reserved	0	Read returns 0. Writes have no effect.
7-0	PSC	0-FFh	Prescale: 8-bit prescaler to divide down the VBUS clock to obtain the I2C module clock. This register must be initialized while the I2C is still in reset (nIRS = 0). The value takes effect on the rising edge of nIRS. See <a href="#">Section 26.1.3</a> for more information.

### 26.6.14 I2C Peripheral ID Register 1 (I2CPID1)

Figure 26-27 and Table 26-23 describe this register.

**Figure 26-27. I2C Peripheral ID Register 1 (I2CPID1) [Offset = 34h]**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

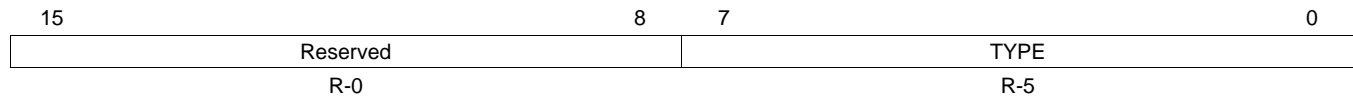
**Table 26-23. I2C Peripheral ID Register 1 (I2CPID1) Field Descriptions**

Bit	Field	Value	Description
15-8	CLASS	0-FFh	Peripheral class: These bits identify the class of peripheral.
7-0	REVISION	0-FFh	Revision level of the I2C: These bits identify the revision level of the I2C.

### 26.6.15 I2C Peripheral ID Register 2 (I2CPID2)

Figure 26-28 and Table 26-24 describe this register.

**Figure 26-28. I2C Peripheral ID Register 2 (I2CPID2) [Offset = 38h]**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

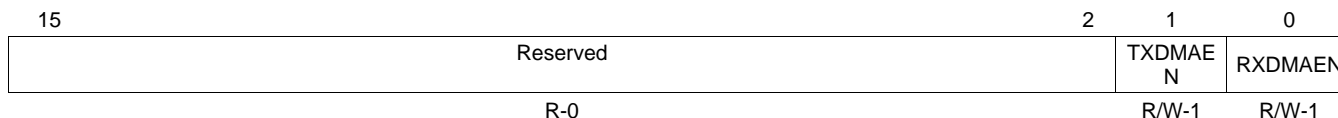
**Table 26-24. I2C Peripheral ID Register 2 (I2CPID2) Field Descriptions**

Bit	Field	Value	Description
15-8	Reserved	0	Read returns 0. Writes have no effect.
7-0	TYPE	0-FFh	Peripheral type: These bits identify the type of peripheral.

### 26.6.16 I2C DMA Control Register (I2CDMACR)

This register contains the transmit and receive DMA enable bits. [Figure 26-29](#) and [Table 26-25](#) describe this register.

**Figure 26-29. I2C DMA Control Register (I2CDMACR) [Offset = 3Ch]**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 26-25. I2C DMA Control Register (I2CDMACR) Field Descriptions**

Bit	Field	Value	Description
15-2	Reserved	0	Read returns 0. Writes have no effect.
1	TXDMAEN	0 1	Transmitter DMA enable: This bit controls the transmit DMA event pin to the system. When this bit is a 1, the DMA transmit event is enabled and the DMA can occur. When this bit is a 0, the DMA transmit event is disabled. Writing a 1 to this bit will send a TXDMA request to the DMA module if PINFUNC is also set to 0. 0 The transmit DMA is disabled. 1 The transmit DMA is enabled.
0	RXDMAEN	0 1	Receive DMA enable: This bit controls the receive DMA event pin to the system. When this bit is 1, the DMA receive event is enabled and the DMA can occur. When this bit is a 0, the DMA receive event is disabled. 0 The receive DMA is disabled. 1 The receive DMA is enabled.

### 26.6.17 I2C Pin Function Register (I2CPFNC)

Figure 26-30 and Table 26-26 describe this register.

**Figure 26-30. I2C Pin Function Register (I2CPFNC) [Offset = 48h]**

15	Reserved	1	0
			PINFUNC
R-0			R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 26-26. I2C Pin Function Register (I2CPFNC) Field Descriptions**

Bit	Field	Value	Description
15-1	Reserved	0	Read returns 0. Writes have no effect.
0	PINFUNC	0	SDA and SCL Pin Function: This bit controls whether the SDA and SCL pins function as I2C pins or as I/O pins. The SDA and SCL pins function as I2C pins.
		1	The SDA and SCL pins function as I/O pins.



### 26.6.18 I2C Pin Direction Register (I2CPDIR)

This register is used to independently configure each I2C pin, when configured as a general purpose I/O, as either an input or output. [Figure 26-31](#) and [Table 26-27](#) describe this register.

**Figure 26-31. I2C Pin Direction Register (I2CPDIR) [Offset = 4Ch]**

15	2	1	0
Reserved		SDADIR	SCLDIR
R-0		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 26-27. I2C Pin Direction Register (I2CPDIR) Field Descriptions**

Bit	Field	Value	Description
15-2	Reserved	0	Read returns 0. Writes have no effect.
1	SDADIR	0	SDA direction: This bit controls the direction of the I2C SDA pin when configured as a GPIO. The SDA pin functions as an input.
		1	SDA pin functions as an output.
0	SCLDIR	0	SCL direction: This bit controls the direction of the I2C SCL pin when configured as a GPIO. The SCL pin functions as an input.
		1	The SCL pin functions as an output.

### 26.6.19 I2C Data Input Register (I2CDIN)

Figure 26-32 and Table 26-28 describe this register.

**Figure 26-32. I2C Data Input Register (I2CDIN) [Offset = 50h]**

15	Reserved	2	1	0
	R-0		SDAIN R-X	SCLIN R-X

LEGEND: R/W = Read/Write; R = Read only; -x = value after reset

**Table 26-28. I2C Data Input Register (I2CDIN) Field Descriptions**

Bit	Field	Value	Description
15-2	Reserved	0	Read returns 0. Writes have no effect.
1	SDAIN		Serial Data In: The value of this bit reflects the value on the SDA pin.
0	SCLIN		Serial clock data in: The value of this bit reflects the value on the SCL pin.

### 26.6.20 I2C Data Output Register (I2CDOUT)

This register contains the values sent to the I2C pins. [Figure 26-33](#) and [Table 26-29](#) describe this register.

**Figure 26-33. I2C Data Output Register (I2CDOUT) [Offset 0x54]**

15	Reserved	2	1	0
	R-0		SDAOUT	SCLOUT
			R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 26-29. I2C Data Output Register (I2CDOUT) Field Descriptions**

Bit	Field	Value	Description
15-2	Reserved	0	Read returns 0. Writes have no effect.
1	SDAOUT		SDA Data Output: This function is only active if the SDA pin is configured as an I/O pin with PINFUNC = 1. This bit contains the value sent to the SDA pin.
		0	The pin is driven low.
		1	The pin is driven high.
0	SCLOUT		SCL Data Output: This function is only active if the SCL pin is configured as an I/O pin with PINFUNC = 1. This bit contains the value sent to the SCL pin.
		0	The pin is driven low.
		1	The pin is driven high.

### 26.6.21 I2C Data Set Register (I2CDSET)

The I2CDSET register is an alias of the I2CDOUT register. When written to at this address, writing a 1 to a bit sets the corresponding bit in I2CDOUT to a 1, while writing a 0 leaves it unchanged. [Figure 26-34](#) and [Table 26-30](#) describe this register.

**Figure 26-34. I2C Data Set Register (I2CDSET) [Offset = 58h]**

15	2	1	0
Reserved		SDASET	SCLSET
R-0		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 26-30. I2C Data Set Register (I2CDSET) Field Descriptions**

Bit	Field	Value	Description
15-2	Reserved	0	Read returns 0. Writes have no effect.
1	SDASET	0	Serial Data Set: This bit is used to set the SDA GPIO pin. <i>Read:</i> Reads return value of SDAOUT. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> Reads return value of SDAOUT. <i>Write:</i> SDAOUT is set to logic high (1).
0	SCLSET	0	Serial Clock Set: This bit is used to set the SCL GPIO pin. <i>Read:</i> Reads return value of SCLOUT. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> Reads return value of SCLOUT. <i>Write:</i> SCLOUT is set to logic high (1).

### 26.6.22 I2C Data Clear Register (I2CDCLR)

The I2CDCLR register is an alias of the I2CDOUT register. When written to at this address, writing a 1 to a bit clears the corresponding bit in I2CDOUT to a 0, while writing a 0 leaves it unchanged. [Figure 26-35](#) and [Table 26-31](#) describe this register.

**Figure 26-35. I2C Data Clear Register (I2CDCLR) [Offset = 5Ch]**

15	2	1	0
Reserved		SDACL	SCLCLR
R-0		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 26-31. I2C Data Clear Register (I2CDSET) Field Descriptions**

Bit	Field	Value	Description
15-2	Reserved	0	Read returns 0. Writes have no effect.
1	SDACL	0	Serial Data Clear: This bit is used to clear the SDA GPIO pin. <i>Read:</i> Reads return value of SDAOUT. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> Reads return value of SDAOUT. <i>Write:</i> SDAOUT is cleared to logic low.
0	SCLCLR	0	Serial Clock Clear: This bit is used to clear the SCL GPIO pin. <i>Read:</i> Reads return value of SCLOUT. <i>Write:</i> Writing a 0 to this bit has no effect.
		1	<i>Read:</i> Reads return value of SCLOUT. <i>Write:</i> SCLOUT is cleared to logic low.

### 26.6.23 I2C Pin Open Drain Register (I2CPDR)

A value of 1 in the I2CPDR register configures the corresponding pin for open drain capability. A value of 0 disables this open drain capability of the pin. [Figure 26-36](#) and [Table 26-32](#) describe this register.

**Figure 26-36. I2C Pin Open Drain Register (I2CPDR) [Offset = 60h]**

15	Reserved	2	1	0
		SDAPDR	SCLPDR	
	R-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 26-32. I2C Pin Open Drain Register (I2CPDR) Field Descriptions**

Bit	Field	Value	Description
15-2	Reserved	0	Read returns 0. Writes have no effect.
1	SDAPDR	0	The open drain function is enabled (the output voltage $V_{OL}$ or lower if SDAOUT = 0 and high-z if SDAOUT = 1).
		1	The open drain function is disabled (output voltage is $V_{OL}$ or lower if SDAOUT = 0 and $V_{OH}$ or higher if SDAOUT=1.)
0	SCLPDR	0	The open drain function is enabled (the output voltage $V_{OL}$ or lower if SCLOUT = 0 and high-z if SCLOUT = 1).
		1	The open drain function is disabled (output voltage is $V_{OL}$ or lower if SCLOUT = 0 and $V_{OH}$ or higher if SCLOUT=1).

### 26.6.24 I2C Pull Disable Register (I2CPDIS)

Values in the I2CPDIS register enable or disable the pull control capability of the pins. A 0 in the I2CPDIS register enables the pull function of the corresponding pin, while a 1 disables the pull function.

Figure 26-37 and Table 26-33 describe this register.

**Figure 26-37. I2C Pull Disable Register (I2CPDIS) [Offset = 64h]**

15	Reserved	2	1	0
			SDAPDIS	SCLPDIS
R-0			R/W-1	R/W-1

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 26-33. I2C Pull Disable Register (I2CDCLR) Field Descriptions**

Bit	Field	Value	Description
15-2	Reserved	0	Read returns 0. Writes have no effect.
1	SDAPDIS	0	SDA pull disable The pull function is enabled.
		1	The pull function is disabled.
0	SCLPDIS	0	SCL pull disable The pull function is enabled.
		1	The pull function is disabled.

### 26.6.25 I2C Pull Select Register (I2CPSEL)

Values in the I2CPSEL register select the pull up or pull down functions of the corresponding pins. A 0 in the I2CPSEL register enables the pull down function of the corresponding pin, while a 1 enables the pull up function of the corresponding pin. [Figure 26-38](#) and [Table 26-34](#) describe this register.

**Figure 26-38. I2C Pull Disable Register (I2CPDIS) [Offset = 68h]**

15	2	1	0
Reserved	SDAPSEL	SCLPSEL	
R-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 26-34. I2C Pull Disable Register (I2CDCLR) Field Descriptions**

Bit	Field	Value	Description
15-2	Reserved	0	Read returns 0. Writes have no effect.
1	SDAPSEL	0	The pull down function is enabled.
		1	The pull up function is enabled.
0	SCLPSEL	0	The pull down function is enabled.
		1	The pull up function is enabled.

#### 26.6.25.1 Summary

The behavior of the input buffer, output buffer, and the pull control is summarized in [Table 26-35](#).

**Table 26-35. Input Buffer, Output Buffer, and Pull Control Behavior as GPIO Pins <sup>(1)</sup>**

Device under Reset?	Pin Direction (DIR) <sup>(2)</sup>	Pull Disable (PULDIS) <sup>(3)</sup>	Pull Select (PULSEL) <sup>(4)</sup>	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

<sup>(1)</sup> 1. X = Don't care

<sup>(2)</sup> 2. DIR = 0 for input, 1 for output

<sup>(3)</sup> 3. PULDIS = 0 for enabling pull control= 1 for disabling pull control

<sup>(4)</sup> 4. PULSEL= 0 for pull-down functionality= 1 for pull-up functionality



### 26.6.26 I2C Pins Slew Rate Select Register (I2CSRS)

This register controls the slew rate of the signal on the I2C pins. [Figure 26-39](#) and [Table 26-36](#) describe this register.

**Figure 26-39. I2C Pins Slew Rate Select Register (I2CSRS) [Offset = 6Ch]**

15	Reserved	2	1	0
R-0			SDASRS	SCLSRSS
			R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 26-36. I2C Pins Slew Rate Select Register (I2CSRS) Field Descriptions**

Bit	Field	Value	Description
15-2	Reserved	0	Read returns 0. Writes have no effect.
1	SDASRS	0	The slow buffer is selected.
		1	The normal buffer is selected.
0	SCLSRSS	0	The slow buffer is selected.
		1	The normal buffer is selected.

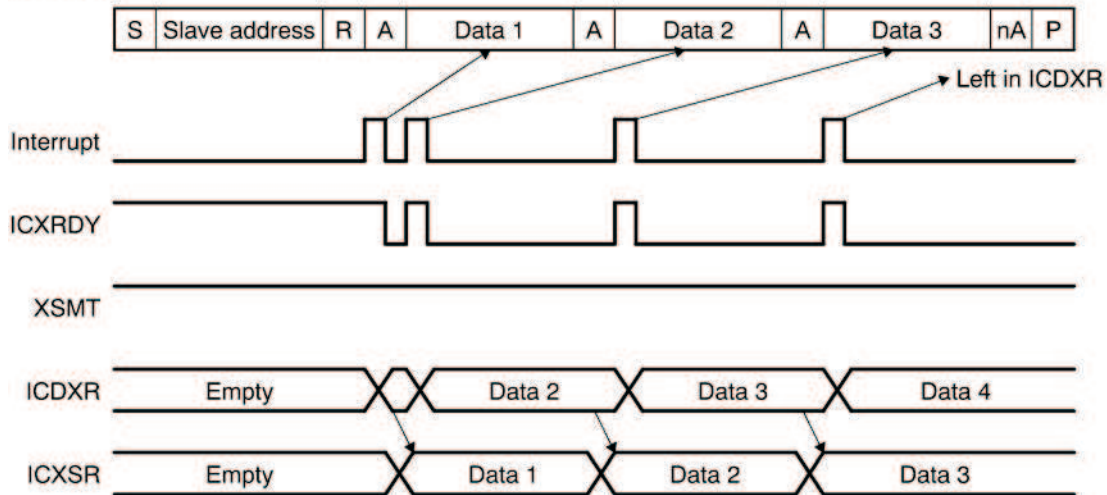
## 26.7 Sample Waveforms

Figure 26-40 provides waveforms to illustrate the difference between normal operation and backward compatibility mode.

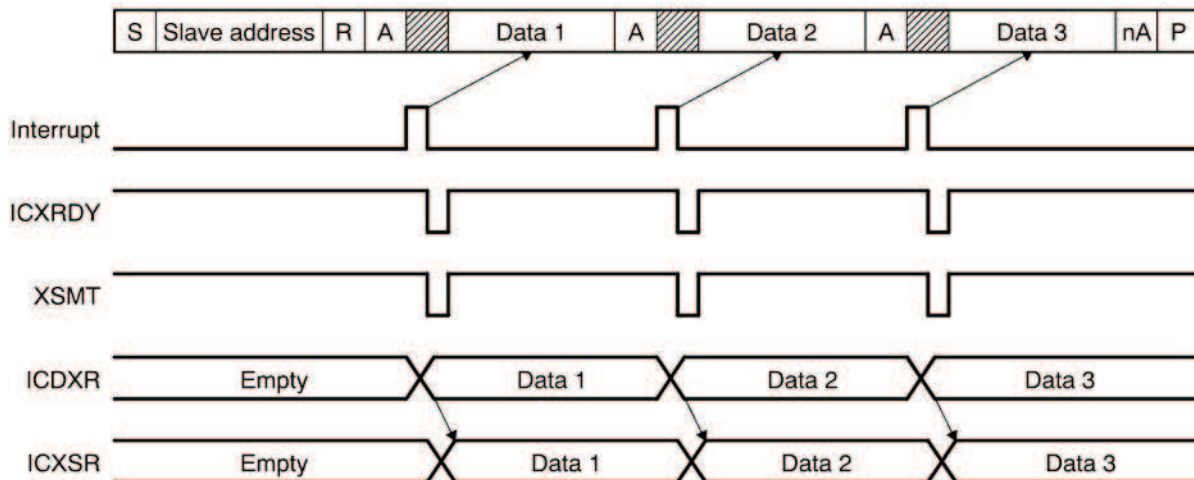
**Figure 26-40. Difference between Normal Operation and Backward Compatibility Mode**

Slave transmitter

a) BCM=1



b) BCM=0





## **EMAC/MDIO Module**

---

---

---

This chapter describes the Ethernet Media Access Controller (EMAC) and physical layer (PHY) device Management Data Input/Output (MDIO) module.

<b>Topic</b>	<b>Page</b>
<b>27.1 Introduction</b> .....	<b>1554</b>
<b>27.2 Architecture</b> .....	<b>1557</b>
<b>27.3 EMAC Control Module Registers</b> .....	<b>1600</b>
<b>27.4 MDIO Registers</b> .....	<b>1614</b>
<b>27.5 EMAC Module Registers</b> .....	<b>1629</b>

## 27.1 Introduction

This document provides a functional description of the Ethernet Media Access Controller (EMAC) and physical layer (PHY) device Management Data Input/Output (MDIO) module integrated in the microcontroller. Included are the features of the EMAC and MDIO modules, a discussion of their architecture and operation, how these modules connect to the outside world, and a description of the registers for each module.

The EMAC controls the flow of packet data from the system to the PHY. The MDIO module controls PHY configuration and status monitoring.

Both the EMAC and the MDIO modules interface to the system core through a custom interface that allows efficient data transmission and reception. This custom interface is referred to as the EMAC control module and is considered integral to the EMAC/MDIO peripheral.

### 27.1.1 Purpose of the Peripheral

The EMAC module is used to move data between the device and another host connected to the same network, in compliance with the Ethernet protocol.

### 27.1.2 Features

The EMAC/MDIO has the following features:

- Synchronous 10/100 Mbps operation.
- Standard Media Independent Interface (MII) and/or Reduced Media Independent Interface (RMII) to physical layer device (PHY).
- EMAC acts as DMA master to either internal or external device memory space.
- Eight receive channels with VLAN tag discrimination for receive quality-of-service (QOS) support.
- Eight transmit channels with round-robin or fixed priority for transmit quality-of-service (QOS) support.
- Ether-Stats and 802.3-Stats statistics gathering.
- Transmit CRC generation selectable on a per channel basis.
- Broadcast frames selection for reception on a single channel.
- Multicast frames selection for reception on a single channel.
- Promiscuous receive mode frames selection for reception on a single channel (all frames, all good frames, short frames, error frames).
- Hardware flow control.
- 8k-byte local EMAC descriptor memory that allows the peripheral to operate on descriptors without affecting the CPU. The descriptor memory holds enough information to transfer up to 512 Ethernet packets without CPU intervention. (This memory is also known as CPPI RAM.)
- Programmable interrupt logic permits the software driver to restrict the generation of back-to-back interrupts, which allows more work to be performed in a single call to the interrupt service routine.

### 27.1.3 Functional Block Diagram

Figure 27-1 shows the three main functional modules of the EMAC/MDIO peripheral:

- EMAC control module
- EMAC module
- MDIO module

The EMAC control module is the main interface between the device core processor to the EMAC and MDIO modules. The EMAC control module controls device interrupts and incorporates an 8k-byte internal RAM to hold EMAC buffer descriptors (also known as CPPI RAM).

The MDIO module implements the 802.3 serial management interface to interrogate and control up to 32 Ethernet PHYs connected to the device by using a shared two-wire bus. Host software uses the MDIO module to configure the autonegotiation parameters of each PHY attached to the EMAC, retrieve the negotiation results, and configure required parameters in the EMAC module for correct operation. The module is designed to allow almost transparent operation of the MDIO interface, with very little maintenance from the core processor.

The EMAC module provides an efficient interface between the processor and the network. The EMAC on this device supports 10Base-T (10 Mbits/sec) and 100BaseTX (100 Mbits/sec), half-duplex and full-duplex mode, and hardware flow control and quality-of-service (QOS) support.

Figure 27-1 shows the main interface between the EMAC control module and the CPU. The following connections are made to the device core:

- The DMA bus connection from the EMAC control module allows the EMAC module to read and write both internal and external memory through the DMA memory transfer controller.
- The EMAC control, EMAC, and MDIO modules all have control registers. These registers are memory-mapped into device memory space. Along with these registers, the control module's internal CPPI RAM is mapped into this same range.
- The EMAC and MDIO interrupts are combined into four interrupt signals within the control module. The Vectored Interrupt Manager (VIM) receives all four interrupt signals from the combiner and submits these interrupt requests to the CPU.

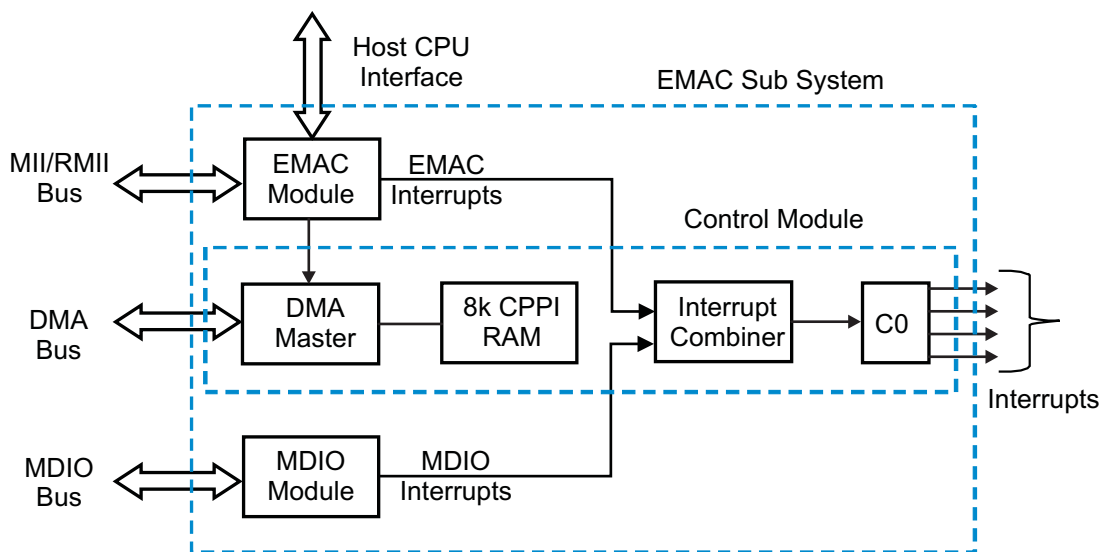


Figure 27-1. EMAC and MDIO Block Diagram

---

### **27.1.4 Industry Standard(s) Compliance Statement**

The EMAC peripheral conforms to the IEEE 802.3 standard, describing the Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer specifications. The IEEE 802.3 standard has also been adopted by ISO/IEC and re-designated as ISO/IEC 8802-3:2000(E).

However, the EMAC deviates from the standard in the way it handles transmit underflow errors. The EMAC MII interface does not use the Transmit Coding Error signal MTXER. Instead of driving the error pin when an underflow condition occurs on a transmitted frame, the EMAC intentionally generates an incorrect checksum by inverting the frame CRC, so that the transmitted frame is detected as an error by the network.

## 27.2 Architecture

This section discusses the architecture and basic function of the EMAC peripheral.

### 27.2.1 Clock Control

All internal EMAC logic is clocked synchronously on the VCLKA4 domain. Please refer to the "Architecture" chapter of the technical reference manual for more details.

The MDIO clock is based on a divide-down of the VCLK3 peripheral bus clock and is specified to run up to 2.5 MHz (although typical operation would be 1.0 MHz). Because the VCLK3 peripheral clock frequency is configurable, the application software or driver must control the divide-down value.

The transmit and receive clock sources are provided by the external PHY to the MII\_TXCLK and MII\_RXCLK pins or to the RMI reference clock pin. Data is transmitted and received with respect to the reference clocks of the interface pins.

The MII interface frequencies for the transmit and receive clocks are fixed by the IEEE 802.3 specification as:

- 2.5 MHz at 10 Mbps
- 25 MHz at 100 Mbps

The RMI interface frequency for the transmit and receive clocks are fixed at 50 MHz for both 10 Mbps and 100 Mbps.

### 27.2.2 Memory Map

The EMAC peripheral includes internal memory that is used to hold buffer descriptions of the Ethernet packets to be received and transmitted. This internal RAM is 2K × 32 bits in size. Data can be written to and read from the EMAC internal memory by either the EMAC or the CPU. It is used to store buffer descriptors that are 4-words (16-bytes) deep. This 8K local memory holds enough information to transfer up to 512 Ethernet packets without CPU intervention. This EMAC RAM is also referred to as the CPPI buffer descriptor memory because it complies with the Communications Port Programming Interface (CPPI) v3.0 standard.

The packet buffer descriptors can also be placed in other on- and off-chip memories such as the CPU RAM. There are some tradeoffs in terms of interconnect bandwidth when descriptors are placed in the CPU RAM, versus when they are placed in the EMAC's internal memory. In general, the EMAC throughput is better when the descriptors are placed in the local EMAC CPPI RAM.

### 27.2.3 Signal Descriptions

The microcontrollers support both the MII and the RMI interfaces. Only one of these two interfaces can be used at a time. A separate control register in the I/O Multiplexing Module (IOMM) allows the application to indicate the actual interface being used. This is the bit 24 of the PINMMR29 control register. This bit is set by default and selects the RMI interface. The application can select the MII interface by clearing this bit. Please refer the I/O Multiplexing Module chapter of the technical reference manual for more details on the procedure to configure the PINMMR registers.

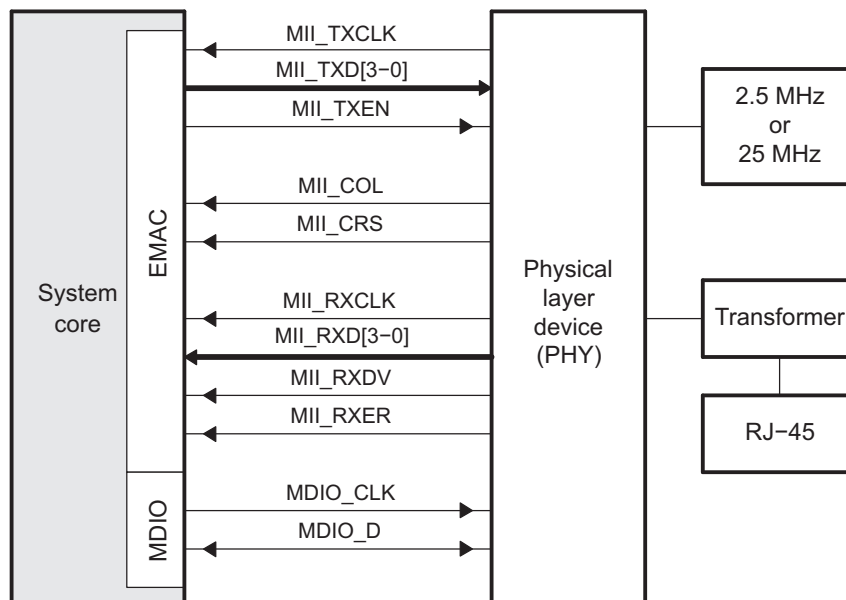
Each of the EMAC and MDIO signals for the MII and RMI interfaces are multiplexed with other I/O functions on this microcontroller. Please refer to [Section 27.2.4](#) for information on configuration of the multiplexing control registers to enable the MII / RMI connections to these I/Os.

#### 27.2.3.1 Media Independent Interface (MII) Connections

[Figure 27-2](#) shows a device with integrated EMAC and MDIO interfaced via a MII connection in a typical system. The EMAC module does not include a transmit error (MTXER) pin. In the case of transmit error, CRC inversion is used to negate the validity of the transmitted frame.

The individual EMAC and MDIO signals for the MII interface are summarized in [Table 27-1](#). For more information, refer to either the IEEE 802.3 standard or ISO/IEC 8802-3:2000(E).



**Figure 27-2. Ethernet Configuration—MII Connections****Table 27-1. EMAC and MDIO Signals for MII Interface**

Signal	Type	Description
MII_TXCLK	I	Transmit clock (MII_TXCLK). The transmit clock is a continuous clock that provides the timing reference for transmit operations. The MII_TXD and MII_TXEN signals are tied to this clock. The clock is generated by the PHY and is 2.5 MHz at 10 Mbps operation and 25 MHz at 100 Mbps operation.
MII_TXD[3-0]	O	Transmit data (MII_TXD). The transmit data pins are a collection of 4 data signals comprising 4 bits of data. MTDX0 is the least-significant bit (LSB). The signals are synchronized by MII_TXCLK and valid only when MII_TXEN is asserted.
MII_TXEN	O	Transmit enable (MII_TXEN). The transmit enable signal indicates that the MII_TXD pins are generating nibble data for use by the PHY. It is driven synchronously to MII_TXCLK.
MII_COL	I	Collision detected (MII_COL). In half-duplex operation, the MII_COL pin is asserted by the PHY when it detects a collision on the network. It remains asserted while the collision condition persists. This signal is not necessarily synchronous to MII_TXCLK nor MII_RXCLK.  In full-duplex operation, the MII_COL pin is used for hardware transmit flow control. Asserting the MII_COL pin will stop packet transmissions; packets in the process of being transmitted when MII_COL is asserted will complete transmission. The MII_COL pin should be held low if hardware transmit flow control is not used.
MII_CRS	I	Carrier sense (MII_CRS). In half-duplex operation, the MII_CRS pin is asserted by the PHY when the network is not idle in either transmit or receive. The pin is deasserted when both transmit and receive are idle. This signal is not necessarily synchronous to MII_TXCLK nor MII_RXCLK.  In full-duplex operation, the MII_CRS pin should be held low.
MII_RXCLK	I	Receive clock (MII_RXCLK). The receive clock is a continuous clock that provides the timing reference for receive operations. The MII_RXD, MII_RXDV, and MII_RXER signals are tied to this clock. The clock is generated by the PHY and is 2.5 MHz at 10 Mbps operation and 25 MHz at 100 Mbps operation.
MII_RXD[3-0]	I	Receive data (MII_RXD). The receive data pins are a collection of 4 data signals comprising 4 bits of data. MRDX0 is the least-significant bit (LSB). The signals are synchronized by MII_RXCLK and valid only when MII_RXDV is asserted.
MII_RXDV	I	Receive data valid (MII_RXDV). The receive data valid signal indicates that the MII_RXD pins are generating nibble data for use by the EMAC. It is driven synchronously to MII_RXCLK.
MII_RXER	I	Receive error (MII_RXER). The receive error signal is asserted for one or more MII_RXCLK periods to indicate that an error was detected in the received frame. This is meaningful only during data reception when MII_RXDV is active.
MDIO_CLK	O	Management data clock (MDIO_CLK). The MDIO data clock is sourced by the MDIO module on the system. It is used to synchronize MDIO data access operations done on the MDIO pin. The frequency of this clock is controlled by the CLKDIV bits in the MDIO control register (CONTROL).

**Table 27-1. EMAC and MDIO Signals for MII Interface (continued)**

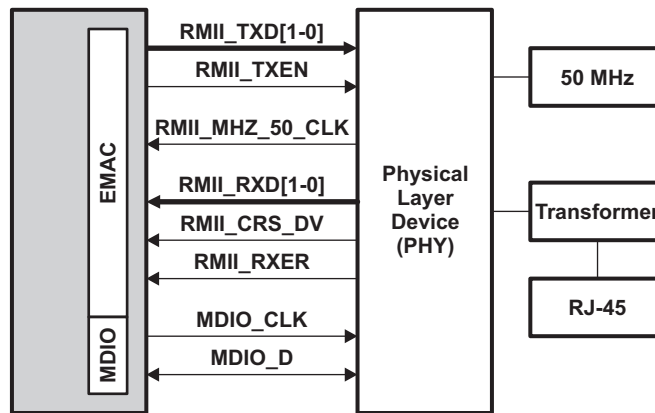
Signal	Type	Description
MDIO_D	I/O	Management data input output (MDIO_D). The MDIO data pin drives PHY management data into and out of the PHY by way of an access frame consisting of start of frame, read/write indication, PHY address, register address, and data bit cycles. The MDIO_D pin acts as an output for all but the data bit cycles at which time it is an input for read operations.

**27.2.3.2 Reduced Media Independent Interface (RMII) Connections**

Figure 27-3 shows a device with integrated EMAC and MDIO interfaced via a RMII connection in a typical system.

The individual EMAC and MDIO signals for the RMII interface are summarized in Table 27-2. For more information, refer to either the IEEE 802.3 standard or ISO/IEC 8802-3:2000(E).

**Figure 27-3. Ethernet Configuration—RMII Connections**



**Table 27-2. EMAC and MDIO Signals for RMII Interface**

Signal	Type	Description
RMI_TXD[1-0]	O	Transmit data (RMI_TXD). The transmit data pins are a collection of 2 bits of data. RMTDX0 is the least-significant bit (LSB). The signals are synchronized by RMI_MHZ_50_CLK and valid only when RMI_TXEN is asserted.
RMI_TXEN	O	Transmit enable (RMI_TXEN). The transmit enable signal indicates that the RMI_TXD pins are generating data for use by the PHY. RMI_TXEN is synchronous to RMI_MHZ_50_CLK.
RMI_MHZ_50_CLK	I	RMII reference clock (RMI_MHZ_50_CLK). The reference clock is used to synchronize all RMII signals. RMI_MHZ_50_CLK must be continuous and fixed at 50 MHz.
RMI_RXD[1-0]	I	Receive data (RMI_RXD). The receive data pins are a collection of 2 bits of data. RMRDX0 is the least-significant bit (LSB). The signals are synchronized by RMI_MHZ_50_CLK and valid only when RMI_CRD_DV is asserted and RMI_RXER is deasserted.
RMI_CRD_DV	I	Carrier sense/receive data valid (RMI_CRD_DV). Multiplexed signal between carrier sense and receive data valid.
RMI_RXER	I	Receive error (RMI_RXER). The receive error signal is asserted to indicate that an error was detected in the received frame.
MDIO_CLK	O	Management data clock (MDIO_CLK). The MDIO data clock is sourced by the MDIO module on the system. It is used to synchronize MDIO data access operations done on the MDIO pin. The frequency of this clock is controlled by the CLKDIV bits in the MDIO control register (CONTROL).
MDIO_D	I/O	Management data input output (MDIO_D). The MDIO data pin drives PHY management data into and out of the PHY by way of an access frame consisting of start of frame, read/write indication, PHY address, register address, and data bit cycles. The MDIO_D pin acts as an output for all but the data bit cycles at which time it is an input for read operations.

### 27.2.4 MII / RMII Signal Multiplexing Control

Each of the MII and RMII interface signals are multiplexed with other functions on this microcontroller. The application must configure the control registers in the I/O multiplexing module in order to enable the MII/RMII functionality on the corresponding I/Os. [Table 27-3](#) shows the byte to be configured to enable the MDIO functions. [Table 27-4](#) shows the byte to be configured to enable the MII or RMII functions. Please refer the I/O Multiplexing Module chapter of the technical reference manual for more details on the procedure to configure the PINMMR registers.

**Table 27-3. MDIO Multiplexing Control**

MDIO Signal Name	Control for Selecting EMAC / MDIO Signal
MDIO_CLK	PINMMR7 [15:8] = 0b00000100
MDIO_D	PINMMR8 [15:8] = 0b00000100

**Table 27-4. MII / RMII Multiplexing Control**

MII / RMII Signal Name	Control for Selecting MII Signal	Control for Selecting RMII Signal
MII_TXCLK	PINMMR17 [7:0] = 0b00000010	-
MII_TXD[3]	PINMMR14 [7:0] = 0b00000100	-
MII_TXD[2]	PINMMR13 [31:24] = 0b00000100	-
MII_TXD[1] / RMII_TXD[1]	PINMMR13 [15:8] = 0b00000100	PINMMR13 [15:8] = 0b00001000
MII_TXD[0] / RMII_TXD[0]	PINMMR13 [7:0] = 0b00000100	PINMMR13 [7:0] = 0b00001000
MII_TXEN / RMII_TXEN	PINMMR13 [23:16] = 0b00000100	PINMMR13 [23:16] = 0b00001000
MII_COL	PINMMR20 [23:16] = 0b00000100	-
MII_CRSDV / RMII_CRSDV	PINMMR17 [23:16] = 0b00000010	PINMMR17 [23:16] = 0b00000100
MII_RXCLK / RMII_50MHZ_CLK	PINMMR14 [15:8] = 0b00000010	PINMMR14 [15:8] = 0b00000100
MII_RXD[3]	PINMMR12 [31:24] = 0b00000100	-
MII_RXD[2]	PINMMR12 [23:16] = 0b00000100	-
MII_RXD[1] / RMII_RXD[1]	PINMMR12 [7:0] = 0b00000010	PINMMR12 [7:0] = 0b00000100
MII_RXD[0] / RMII_RXD[0]	PINMMR11 [31:24] = 0b00000100	PINMMR11 [31:24] = 0b00001000
MII_RXDV	PINMMR19 [15:8] = 0b00000010	-
MII_RXER / RMII_RXER	PINMMR10 [7:0] = 0b00000010	PINMMR10 [7:0] = 0b00000100

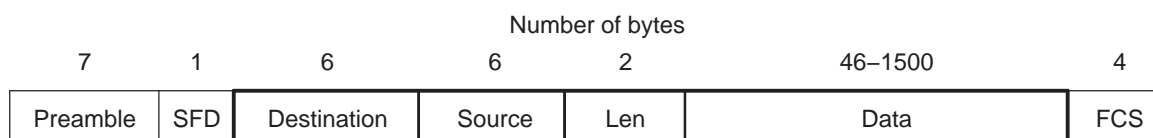
### 27.2.5 Ethernet Protocol Overview

A brief overview of the Ethernet protocol is given in the following subsections. See the IEEE 802.3 standard document for in-depth information on the Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method.

#### 27.2.5.1 Ethernet Frame Format

All the Ethernet technologies use the same frame structure. The format of an Ethernet frame is shown in [Figure 27-4](#) and described in [Table 27-5](#). The Ethernet packet, which is the collection of bytes representing the data portion of a single Ethernet frame on the wire, is shown outlined in bold. The Ethernet frames are of variable lengths, with no frame smaller than 64 bytes or larger than RXMAXLEN bytes (header, data, and CRC).

**Figure 27-4. Ethernet Frame Format**



Legend: SFD=Start Frame Delimeter; FCS=Frame Check Sequence (CRC)

**Table 27-5. Ethernet Frame Description**

<b>Field</b>	<b>Bytes</b>	<b>Description</b>
Preamble	7	Preamble. These 7 bytes have a fixed value of 55h and serve to wake up the receiving EMAC ports and to synchronize their clocks to that of the sender's clock.
SFD	1	Start of Frame Delimiter. This field with a value of 5Dh immediately follows the preamble pattern and indicates the start of important data.
Destination	6	Destination address. This field contains the Ethernet MAC address of the EMAC port for which the frame is intended. It may be an individual or multicast (including broadcast) address. When the destination EMAC port receives an Ethernet frame with a destination address that does not match any of its MAC physical addresses, and no promiscuous, multicast or broadcast channel is enabled, it discards the frame.
Source	6	Source address. This field contains the MAC address of the Ethernet port that transmits the frame to the Local Area Network.
Len	2	Length/Type field. The length field indicates the number of EMAC client data bytes contained in the subsequent data field of the frame. This field can also be used to identify the type of data the frame is carrying.
Data	46 to (RXMAXLEN - 18)	Data field. This field carries the datagram containing the upper layer protocol frame, that is, IP layer datagram. The maximum transfer unit (MTU) of Ethernet is (RXMAXLEN - 18) bytes. This means that if the upper layer protocol datagram exceeds (RXMAXLEN - 18) bytes, then the host has to fragment the datagram and send it in multiple Ethernet packets. The minimum size of the data field is 46 bytes. This means that if the upper layer datagram is less than 46 bytes, the data field has to be extended to 46 bytes by appending extra bits after the data field, but prior to calculating and appending the FCS.
FCS	4	Frame Check Sequence. A cyclic redundancy check (CRC) is used by the transmit and receive algorithms to generate a CRC value for the FCS field. The frame check sequence covers the 60 to 1514 bytes of the packet data. Note that this 4-byte field may or may not be included as part of the packet data, depending on how the EMAC is configured.

### 27.2.5.2 Ethernet's Multiple Access Protocol

Nodes in an Ethernet Local Area Network are interconnected by a broadcast channel -- when an EMAC port transmits a frame, all the adapters on the local network receive the frame. Carrier Sense Multiple Access with Collision Detection (CSMA/CD) algorithms are used when the EMAC operates in half-duplex mode. When operating in full-duplex mode, there is no contention for use of a shared medium because there are exactly two ports on the local network.

Each port runs the CSMA/CD protocol without explicit coordination with the other ports on the Ethernet network. Within a specific port, the CSMA/CD protocol works as follows:

1. The port obtains data from upper layer protocols at its node, prepares an Ethernet frame, and puts the frame in a buffer.
2. If the port senses that the medium is idle, it starts to transmit the frame. If the port senses that the transmission medium is busy, it waits until it no longer senses energy (plus an Inter-Packet Gap time) and then starts to transmit the frame.
3. While transmitting, the port monitors for the presence of signal energy coming from other ports. If the port transmits the entire frame without detecting signal energy from other Ethernet devices, the port is done with the frame.
4. If the port detects signal energy from other ports while transmitting, it stops transmitting its frame and instead transmits a 48-bit jam signal.
5. After transmitting the jam signal, the port enters an exponential backoff phase. If a data frame encounters back-to-back collisions, the port chooses a random value that is dependent on the number of collisions. The port then waits an amount of time that is a multiple of this random value and returns to step 2.

### 27.2.6 Programming Interface

#### 27.2.6.1 Packet Buffer Descriptors

The buffer descriptor is a central part of the EMAC module and is how the application software describes Ethernet packets to be sent and empty buffers to be filled with incoming packet data. The basic descriptor format is shown in [Figure 27-5](#) and described in [Table 27-6](#).

For example, consider three packets to be transmitted: Packet A is a single fragment (60 bytes), Packet B is fragmented over three buffers (1514 bytes total), and Packet C is a single fragment (1514 bytes). The linked list of descriptors to describe these three packets is shown in [Figure 27-6](#).

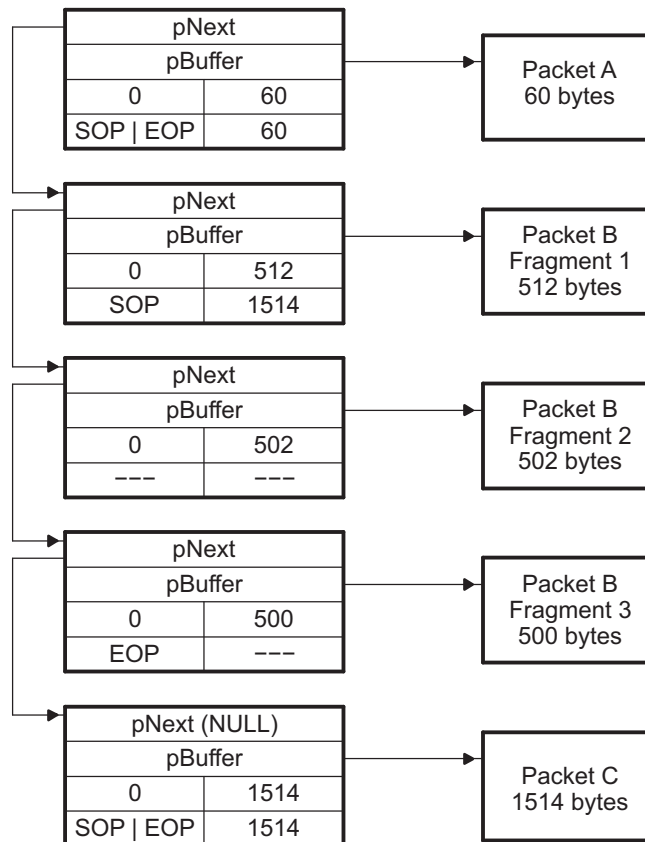
**Figure 27-5. Basic Descriptor Format**

Word Offset	Bit Fields			
	31	16	15	0
0	Next Descriptor Pointer			
1	Buffer Pointer			
2	Buffer Offset		Buffer Length	
3	Flags		Packet Length	

**Table 27-6. Basic Descriptor Description**

Word Offset	Field	Field Description
0	Next Descriptor Pointer	The next descriptor pointer is used to create a single linked list of descriptors. Each descriptor describes a packet or a packet fragment. When a descriptor points to a single buffer packet or the first fragment of a packet, the start of packet (SOP) flag is set in the flags field. When a descriptor points to a single buffer packet or the last fragment of a packet, the end of packet (EOP) flag is set. When a packet is fragmented, each fragment must have its own descriptor and appear sequentially in the descriptor linked list.
1	Buffer Pointer	The buffer pointer refers to the actual memory buffer that contains packet data during transmit operations, or is an empty buffer ready to receive packet data during receive operations.
2	Buffer Offset	The buffer offset is the offset from the start of the packet buffer to the first byte of valid data. This field only has meaning when the buffer descriptor points to a buffer that actually contains data.
	Buffer Length	The buffer length is the actual number of valid packet data bytes stored in the buffer. If the buffer is empty and waiting to receive data, this field represents the size of the empty buffer.
3	Flags	The flags field contains more information about the buffer, such as, is it the first fragment in a packet (SOP), the last fragment in a packet (EOP), or contains an entire contiguous Ethernet packet (both SOP and EOP). The flags are described in <a href="#">Section 27.2.6.4</a> and <a href="#">Section 27.2.6.5</a> .
	Packet Length	The packet length only has meaning for buffers that both contain data and are the start of a new packet (SOP). In the case of SOP descriptors, the packet length field contains the length of the entire Ethernet packet, regardless if it is contained in a single buffer or fragmented over several buffers.

**Figure 27-6. Typical Descriptor Linked List**



### 27.2.6.2 Transmit and Receive Descriptor Queues

The EMAC module processes descriptors in linked lists as discussed in [Section 27.2.6.1](#). The lists used by the EMAC are maintained by the application software through the use of the head descriptor pointer registers (HDP). The EMAC supports eight channels for transmit and eight channels for receive. The corresponding head descriptor pointers are:

- TX $n$ HDP - Transmit Channel  $n$  DMA Head Descriptor Pointer Register
- RX $n$ HDP - Receive Channel  $n$  DMA Head Descriptor Pointer Register

After an EMAC reset and before enabling the EMAC for send and receive, all 16 head descriptor pointer registers must be initialized to 0.

The EMAC uses a simple system to determine if a descriptor is currently owned by the EMAC or by the application software. There is a flag in the buffer descriptor flags called OWNER. When this flag is set, the packet that is referenced by the descriptor is considered to be owned by the EMAC. Note that ownership is done on a packet based granularity, not on descriptor granularity, so only SOP descriptors make use of the OWNER flag. As packets are processed, the EMAC patches the SOP descriptor of the corresponding packet and clears the OWNER flag. This is an indication that the EMAC has finished processing all descriptors up to and including the first with the EOP flag set, indicating the end of the packet (note this may only be one descriptor with both the SOP and EOP flags set).

To add a descriptor or a linked list of descriptors to an EMAC descriptor queue for the first time, the software application simply writes the pointer to the descriptor or first descriptor of a list to the corresponding HDP register. Note that the last descriptor in the list must have its “next” pointer cleared to 0. This is the only way the EMAC has of detecting the end of the list. Therefore, in the case where only a single descriptor is added, its “next descriptor” pointer must be initialized to 0.

The HDP must never be written to while a list is active. To add additional descriptors to a descriptor list already owned by the EMAC, the NULL “next” pointer of the last descriptor of the previous list is patched with a pointer to the first descriptor of the new list. The list of new descriptors to be appended to the existing list must itself be NULL terminated before the pointer patch is performed.

There is a potential race condition where the EMAC may read the “next” pointer of a descriptor as NULL in the instant before an application appends additional descriptors to the list by patching the pointer. This case is handled by the software application always examining the buffer descriptor flags of all EOP packets, looking for a special flag called end of queue (EOQ). The EOQ flag is set by the EMAC on the last descriptor of a packet when the descriptor’s “next” pointer is NULL. This is the way the EMAC indicates to the software application that it believes it has reached the end of the list. When the software application sees the EOQ flag set, the application may at that time submit the new list, or the portion of the appended list that was missed by writing the new list pointer to the same HDP that started the process.

This process applies when adding packets to a transmit list, and empty buffers to a receive list.



### 27.2.6.3 Transmit and Receive EMAC Interrupts

The EMAC processes descriptors in linked list chains as discussed in [Section 27.2.6.1](#), using the linked list queue mechanism discussed in [Section 27.2.6.2](#).

The EMAC synchronizes descriptor list processing through the use of interrupts to the software application. The interrupts are controlled by the application using the interrupt masks, global interrupt enable, and the completion pointer register (CP). The CP is also called the interrupt acknowledge register.

The EMAC supports eight channels for transmit and eight channels for receive. The corresponding completion pointer registers are:

- TX $n$ CP - Transmit Channel  $n$  Completion Pointer (Interrupt Acknowledge) Register
- RX $n$ CP - Receive Channel  $n$  Completion Pointer (Interrupt Acknowledge) Register

These registers serve two purposes. When read, they return the pointer to the last descriptor that the EMAC has processed. When written by the software application, the value represents the last descriptor processed by the software application. When these two values do not match, the interrupt is active.

Interrupts in the EMAC control module are routed to the Vectored Interrupt Manager (VIM) as four separate interrupt requests. The interrupt configuration determines whether or not an active interrupt request actually interrupts the CPU. In general the following settings are required for basic EMAC transmit and receive interrupts:

1. EMAC transmit and receive interrupts are enabled by setting the mask registers RXINTMASKSET and TXINTMASKSET
2. Global interrupts are set in the EMAC control module: C0RXEN and C0TXEN
3. The VIM is configured to accept C0\_RX\_PULSE and C0\_TX\_PULSE interrupts from the EMAC control module
4. The normal mode (IRQ) interrupts are enabled in the Cortex-R4F CPU

Whether or not the interrupt is enabled, the current state of the receive or transmit channel interrupt can be examined directly by the software application reading the EMAC receive interrupt status (unmasked) register (RXINTSTATRAW) and transmit interrupt status (unmasked) register (TXINTSTATRAW).

After servicing transmit or receive interrupts, the application software must acknowledge both the EMAC and EMAC control module interrupts.

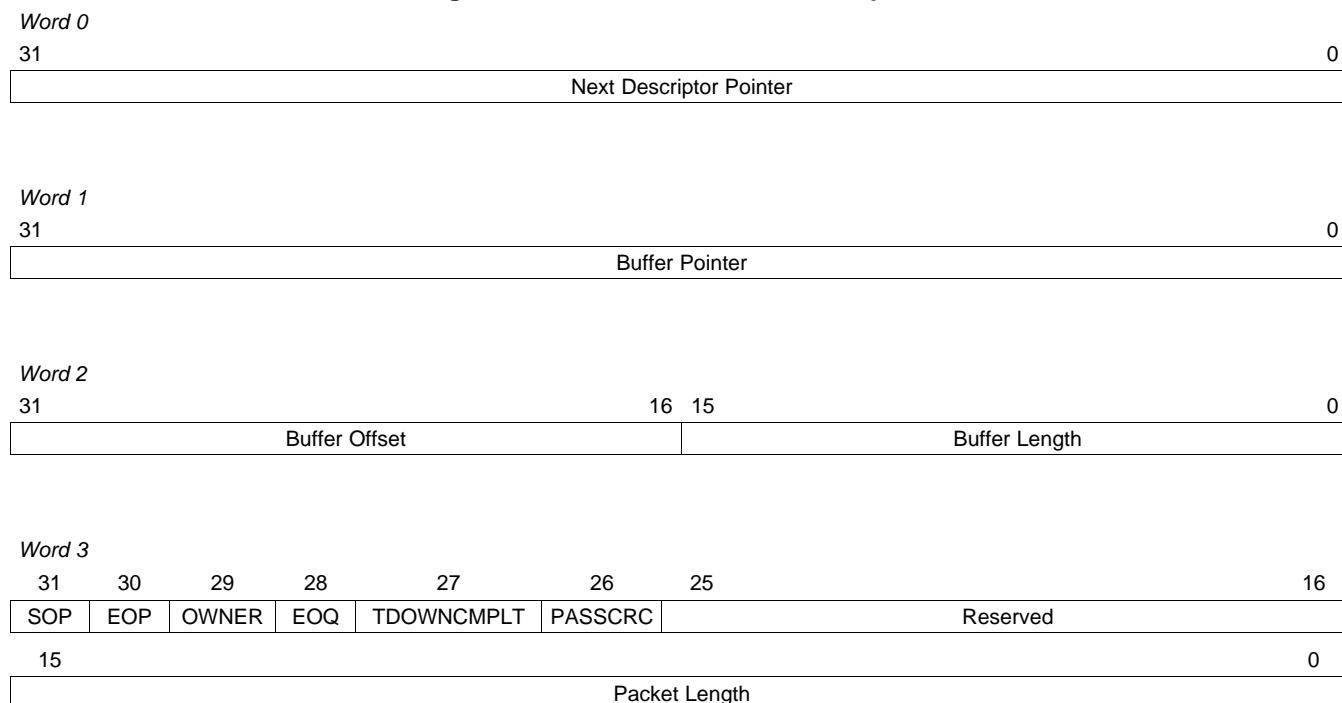
EMAC interrupts are acknowledged when the application software updates the value of TX $n$ CP or RX $n$ CP with a value that matches the internal value kept by the EMAC. This mechanism ensures that the application software never misses an EMAC interrupt because the interrupt acknowledgment is tied directly to the buffer descriptor processing.

EMAC control module interrupts are acknowledged when the application software writes the appropriate C0TX or C0RX key to the EMAC End-Of-Interrupt Vector register (MACEOIVECTOR). The MACEOIVECTOR behaves as an interrupt pulse interlock -- once the EMAC control module has issued an interrupt pulse to the CPU, it will not generate further pulses of the same type until the original pulse has been acknowledged.

### 27.2.6.4 Transmit Buffer Descriptor Format

A transmit (TX) buffer descriptor ([Figure 27-7](#)) is a contiguous block of four 32-bit data words aligned on a 32-bit boundary that describes a packet or a packet fragment. [Example 27-1](#) shows the transmit buffer descriptor described by a C structure.



**Figure 27-7. Transmit Buffer Descriptor Format****Example 27-1. Transmit Buffer Descriptor in C Structure Format**

```

/*
// EMAC Descriptor
//
// The following is the format of a single buffer descriptor
// on the EMAC.
*/
typedef struct _EMAC_Desc {
    struct _EMAC_Desc *pNext; /* Pointer to next descriptor in chain */
    Uint8 *pBuffer; /* Pointer to data buffer */
    Uint32 BufOffLen; /* Buffer Offset(MSW) and Length(LSW) */
    Uint32 PktFlgLen; /* Packet Flags(MSW) and Length(LSW) */
} EMAC_Desc;

/* Packet Flags */
#define EMAC_DSC_FLAG_SOP 0x80000000u
#define EMAC_DSC_FLAG_EOP 0x40000000u
#define EMAC_DSC_FLAG_OWNER 0x20000000u
#define EMAC_DSC_FLAG_EOQ 0x10000000u
#define EMAC_DSC_FLAG_TDOWNCMPLT 0x08000000u
#define EMAC_DSC_FLAG_PASSCRC 0x04000000u

```

#### **27.2.6.4.1 Next Descriptor Pointer**

The next descriptor pointer points to the 32-bit word aligned memory address of the next buffer descriptor in the transmit queue. This pointer is used to create a linked list of buffer descriptors. If the value of this pointer is zero, then the current buffer is the last buffer in the queue. The software application must set this value prior to adding the descriptor to the active transmit list. This pointer is not altered by the EMAC.

The value of pNext should never be altered once the descriptor is in an active transmit queue, unless its current value is NULL. If the pNext pointer is initially NULL, and more packets need to be queued for transmit, the software application may alter this pointer to point to a newly appended descriptor. The EMAC will use the new pointer value and proceed to the next descriptor unless the pNext value has already been read. In this latter case, the transmitter will halt on the transmit channel in question, and the software application may restart it at that time. The software can detect this case by checking for an end of queue (EOQ) condition flag on the updated packet descriptor when it is returned by the EMAC.

#### **27.2.6.4.2 Buffer Pointer**

The buffer pointer is the byte-aligned memory address of the memory buffer associated with the buffer descriptor. The software application must set this value prior to adding the descriptor to the active transmit list. This pointer is not altered by the EMAC.

#### **27.2.6.4.3 Buffer Offset**

This 16-bit field indicates how many unused bytes are at the start of the buffer. For example, a value of 0000h indicates that no unused bytes are at the start of the buffer and that valid data begins on the first byte of the buffer, while a value of 000Fh indicates that the first 15 bytes of the buffer are to be ignored by the EMAC and that valid buffer data starts on byte 16 of the buffer. The software application must set this value prior to adding the descriptor to the active transmit list. This field is not altered by the EMAC.

Note that this value is only checked on the first descriptor of a given packet (where the start of packet (SOP) flag is set). It can not be used to specify the offset of subsequent packet fragments. Also, since the buffer pointer may point to any byte-aligned address, this field may be entirely superfluous, depending on the device driver architecture.

The range of legal values for this field is 0 to (Buffer Length – 1).

#### **27.2.6.4.4 Buffer Length**

This 16-bit field indicates how many valid data bytes are in the buffer. On single fragment packets, this value is also the total length of the packet data to be transmitted. If the buffer offset field is used, the offset bytes are not counted as part of this length. This length counts only valid data bytes. The software application must set this value prior to adding the descriptor to the active transmit list. This field is not altered by the EMAC.

#### **27.2.6.4.5 Packet Length**

This 16-bit field specifies the number of data bytes in the entire packet. Any leading buffer offset bytes are not included. The sum of the buffer length fields of each of the packet's fragments (if more than one) must be equal to the packet length. The software application must set this value prior to adding the descriptor to the active transmit list. This field is not altered by the EMAC. This value is only checked on the first descriptor of a given packet (where the start of packet (SOP) flag is set).

#### **27.2.6.4.6 Start of Packet (SOP) Flag**

When set, this flag indicates that the descriptor points to a packet buffer that is the start of a new packet. In the case of a single fragment packet, both the SOP and end of packet (EOP) flags are set. Otherwise, the descriptor pointing to the last packet buffer for the packet sets the EOP flag. This bit is set by the software application and is not altered by the EMAC.

#### **27.2.6.4.7 End of Packet (EOP) Flag**

When set, this flag indicates that the descriptor points to a packet buffer that is last for a given packet. In the case of a single fragment packet, both the start of packet (SOP) and EOP flags are set. Otherwise, the descriptor pointing to the last packet buffer for the packet sets the EOP flag. This bit is set by the software application and is not altered by the EMAC.

#### **27.2.6.4.8 Ownership (OWNER) Flag**

When set, this flag indicates that all the descriptors for the given packet (from SOP to EOP) are currently owned by the EMAC. This flag is set by the software application on the SOP packet descriptor before adding the descriptor to the transmit descriptor queue. For a single fragment packet, the SOP, EOP, and OWNER flags are all set. The OWNER flag is cleared by the EMAC once it is finished with all the descriptors for the given packet. Note that this flag is valid on SOP descriptors only.

#### **27.2.6.4.9 End of Queue (EOQ) Flag**

When set, this flag indicates that the descriptor in question was the last descriptor in the transmit queue for a given transmit channel, and that the transmitter has halted. This flag is initially cleared by the software application prior to adding the descriptor to the transmit queue. This bit is set by the EMAC when the EMAC identifies that a descriptor is the last for a given packet (the EOP flag is set), and there are no more descriptors in the transmit list (next descriptor pointer is NULL).

The software application can use this bit to detect when the EMAC transmitter for the corresponding channel has halted. This is useful when the application appends additional packet descriptors to a transmit queue list that is already owned by the EMAC. Note that this flag is valid on EOP descriptors only.

#### **27.2.6.4.10 Teardown Complete (TDOWNCMPLT) Flag**

This flag is used when a transmit queue is being torn down, or aborted, instead of allowing it to be transmitted. This would happen under device driver reset or shutdown conditions. The EMAC sets this bit in the SOP descriptor of each packet as it is aborted from transmission.

Note that this flag is valid on SOP descriptors only. Also note that only the first packet in an unsent list has the TDOWNCMPLT flag set. Subsequent descriptors are not processed by the EMAC.

#### **27.2.6.4.11 Pass CRC (PASSCRC) Flag**

This flag is set by the software application in the SOP packet descriptor before it adds the descriptor to the transmit queue. Setting this bit indicates to the EMAC that the 4 byte Ethernet CRC is already present in the packet data, and that the EMAC should not generate its own version of the CRC.

When the CRC flag is cleared, the EMAC generates and appends the 4-byte CRC. The buffer length and packet length fields do not include the CRC bytes. When the CRC flag is set, the 4-byte CRC is supplied by the software application and is already appended to the end of the packet data. The buffer length and packet length fields include the CRC bytes, as they are part of the valid packet data. Note that this flag is valid on SOP descriptors only.

### 27.2.6.5 Receive Buffer Descriptor Format

A receive (RX) buffer descriptor (Figure 27-8) is a contiguous block of four 32-bit data words aligned on a 32-bit boundary that describes a packet or a packet fragment. Example 27-2 shows the receive buffer descriptor described by a C structure.

#### 27.2.6.5.1 Next Descriptor Pointer

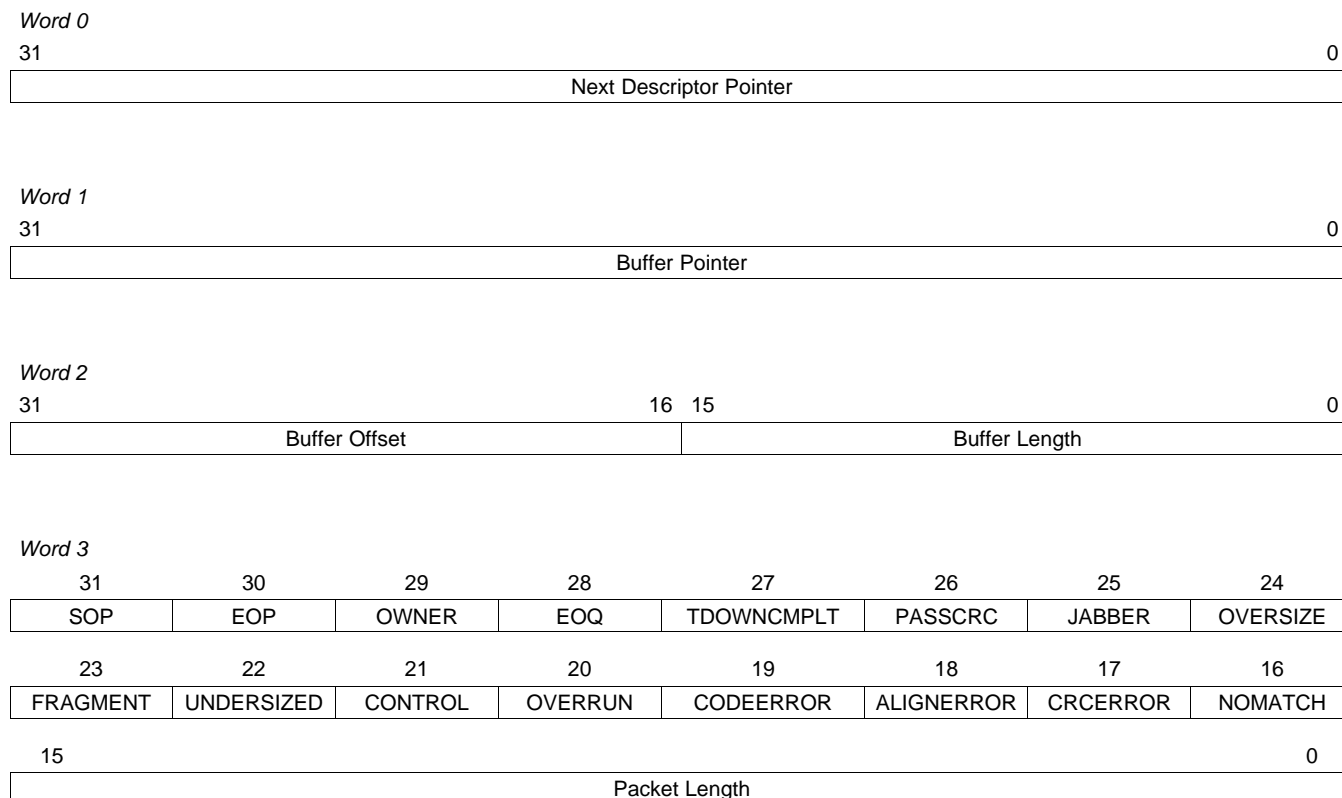
This pointer points to the 32-bit word aligned memory address of the next buffer descriptor in the receive queue. This pointer is used to create a linked list of buffer descriptors. If the value of this pointer is zero, then the current buffer is the last buffer in the queue. The software application must set this value prior to adding the descriptor to the active receive list. This pointer is not altered by the EMAC.

The value of pNext should never be altered once the descriptor is in an active receive queue, unless its current value is NULL. If the pNext pointer is initially NULL, and more empty buffers can be added to the pool, the software application may alter this pointer to point to a newly appended descriptor. The EMAC will use the new pointer value and proceed to the next descriptor unless the pNext value has already been read. In this latter case, the receiver will halt the receive channel in question, and the software application may restart it at that time. The software can detect this case by checking for an end of queue (EOQ) condition flag on the updated packet descriptor when it is returned by the EMAC.

#### 27.2.6.5.2 Buffer Pointer

The buffer pointer is the byte-aligned memory address of the memory buffer associated with the buffer descriptor. The software application must set this value prior to adding the descriptor to the active receive list. This pointer is not altered by the EMAC.

Figure 27-8. Receive Buffer Descriptor Format



**Example 27-2. Receive Buffer Descriptor in C Structure Format**

```

/*
// EMAC Descriptor
//
// The following is the format of a single buffer descriptor
// on the EMAC.
*/
typedef struct _EMAC_Desc {
    struct _EMAC_Desc *pNext; /* Pointer to next descriptor in chain */
    Uint8 *pBuffer; /* Pointer to data buffer */
    Uint32 BufOffLen; /* Buffer Offset(MSW) and Length(LSW) */
    Uint32 PktFlgLen; /* Packet Flags(MSW) and Length(LSW) */
} EMAC_Desc;

/* Packet Flags */
#define EMAC_DSC_FLAG_SOP 0x80000000u
#define EMAC_DSC_FLAG_EOP 0x40000000u
#define EMAC_DSC_FLAG_OWNER 0x20000000u
#define EMAC_DSC_FLAG_EOQ 0x10000000u
#define EMAC_DSC_FLAG_TDOWNCMPLT 0x08000000u
#define EMAC_DSC_FLAG_PASSCRC 0x04000000u
#define EMAC_DSC_FLAG_JABBER 0x02000000u
#define EMAC_DSC_FLAG_OVERSIZE 0x01000000u
#define EMAC_DSC_FLAG_FRAGMENT 0x00800000u
#define EMAC_DSC_FLAG_UNDERSIZED 0x00400000u
#define EMAC_DSC_FLAG_CONTROL 0x00200000u
#define EMAC_DSC_FLAG_OVERRUN 0x00100000u
#define EMAC_DSC_FLAG_CODEERROR 0x00080000u
#define EMAC_DSC_FLAG_ALIGNERROR 0x00040000u
#define EMAC_DSC_FLAG_CRCERROR 0x00020000u
#define EMAC_DSC_FLAG_NOMATCH 0x00010000u

```

**27.2.6.5.3 Buffer Offset**

This 16-bit field must be initialized to zero by the software application before adding the descriptor to a receive queue.

Whether or not this field is updated depends on the setting of the RXBUFFEROFFSET register. When the offset register is set to a non-zero value, the received packet is written to the packet buffer at an offset given by the value of the register, and this value is also written to the buffer offset field of the descriptor.

When a packet is fragmented over multiple buffers because it does not fit in the first buffer supplied, the buffer offset only applies to the first buffer in the list, which is where the start of packet (SOP) flag is set in the corresponding buffer descriptor. In other words, the buffer offset field is only updated by the EMAC on SOP descriptors.

The range of legal values for the BUFFEROFFSET register is 0 to (Buffer Length – 1) for the smallest value of buffer length for all descriptors in the list.

#### **27.2.6.5.4 Buffer Length**

This 16-bit field is used for two purposes:

- Before the descriptor is first placed on the receive queue by the application software, the buffer length field is first initialized by the software to have the physical size of the empty data buffer pointed to by the buffer pointer field.
- After the empty buffer has been processed by the EMAC and filled with received data bytes, the buffer length field is updated by the EMAC to reflect the actual number of valid data bytes written to the buffer.

#### **27.2.6.5.5 Packet Length**

This 16-bit field specifies the number of data bytes in the entire packet. This value is initialized to zero by the software application for empty packet buffers. The value is filled in by the EMAC on the first buffer used for a given packet. This is signified by the EMAC setting a start of packet (SOP) flag. The packet length is set by the EMAC on all SOP buffer descriptors.

#### **27.2.6.5.6 Start of Packet (SOP) Flag**

When set, this flag indicates that the descriptor points to a packet buffer that is the start of a new packet. In the case of a single fragment packet, both the SOP and end of packet (EOP) flags are set. Otherwise, the descriptor pointing to the last packet buffer for the packet has the EOP flag set. This flag is initially cleared by the software application before adding the descriptor to the receive queue. This bit is set by the EMAC on SOP descriptors.

#### **27.2.6.5.7 End of Packet (EOP) Flag**

When set, this flag indicates that the descriptor points to a packet buffer that is last for a given packet. In the case of a single fragment packet, both the start of packet (SOP) and EOP flags are set. Otherwise, the descriptor pointing to the last packet buffer for the packet has the EOP flag set. This flag is initially cleared by the software application before adding the descriptor to the receive queue. This bit is set by the EMAC on EOP descriptors.

#### **27.2.6.5.8 Ownership (OWNER) Flag**

When set, this flag indicates that the descriptor is currently owned by the EMAC. This flag is set by the software application before adding the descriptor to the receive descriptor queue. This flag is cleared by the EMAC once it is finished with a given set of descriptors, associated with a received packet. The flag is updated by the EMAC on SOP descriptor only. So when the application identifies that the OWNER flag is cleared on an SOP descriptor, it may assume that all descriptors up to and including the first with the EOP flag set have been released by the EMAC. (Note that in the case of single buffer packets, the same descriptor will have both the SOP and EOP flags set.)

#### **27.2.6.5.9 End of Queue (EOQ) Flag**

When set, this flag indicates that the descriptor in question was the last descriptor in the receive queue for a given receive channel, and that the corresponding receiver channel has halted. This flag is initially cleared by the software application prior to adding the descriptor to the receive queue. This bit is set by the EMAC when the EMAC identifies that a descriptor is the last for a given packet received (also sets the EOP flag), and there are no more descriptors in the receive list (next descriptor pointer is NULL).

The software application can use this bit to detect when the EMAC receiver for the corresponding channel has halted. This is useful when the application appends additional free buffer descriptors to an active receive queue. Note that this flag is valid on EOP descriptors only.

#### **27.2.6.5.10 Teardown Complete (TDOWNCMPLT) Flag**

This flag is used when a receive queue is being torn down, or aborted, instead of being filled with received data. This would happen under device driver reset or shutdown conditions. The EMAC sets this bit in the descriptor of the first free buffer when the tear down occurs. No additional queue processing is performed.

**27.2.6.5.11 Pass CRC (PASSCRC) Flag**

This flag is set by the EMAC in the SOP buffer descriptor if the received packet includes the 4-byte CRC. This flag should be cleared by the software application before submitting the descriptor to the receive queue.

**27.2.6.5.12 Jabber Flag**

This flag is set by the EMAC in the SOP buffer descriptor, if the received packet is a jabber frame and was not discarded because the RXCEFEN bit was set in the RXMBPENABLE. Jabber frames are frames that exceed the RXMAXLEN in length, and have CRC, code, or alignment errors.

**27.2.6.5.13 Oversize Flag**

This flag is set by the EMAC in the SOP buffer descriptor, if the received packet is an oversized frame and was not discarded because the RXCEFEN bit was set in the RXMBPENABLE.

**27.2.6.5.14 Fragment Flag**

This flag is set by the EMAC in the SOP buffer descriptor, if the received packet is only a packet fragment and was not discarded because the RXCEFEN bit was set in the RXMBPENABLE.

**27.2.6.5.15 Undersized Flag**

This flag is set by the EMAC in the SOP buffer descriptor, if the received packet is undersized and was not discarded because the RXCSFEN bit was set in the RXMBPENABLE.

**27.2.6.5.16 Control Flag**

This flag is set by the EMAC in the SOP buffer descriptor, if the received packet is an EMAC control frame and was not discarded because the RXCMFEN bit was set in the RXMBPENABLE.

**27.2.6.5.17 Overrun Flag**

This flag is set by the EMAC in the SOP buffer descriptor, if the received packet was aborted due to a receive overrun.

**27.2.6.5.18 Code Error (CODEERROR) Flag**

This flag is set by the EMAC in the SOP buffer descriptor, if the received packet contained a code error and was not discarded because the RXCEFEN bit was set in the RXMBPENABLE.

**27.2.6.5.19 Alignment Error (ALIGNERROR) Flag**

This flag is set by the EMAC in the SOP buffer descriptor, if the received packet contained an alignment error and was not discarded because the RXCEFEN bit was set in the RXMBPENABLE.

**27.2.6.5.20 CRC Error (CRCERROR) Flag**

This flag is set by the EMAC in the SOP buffer descriptor, if the received packet contained a CRC error and was not discarded because the RXCEFEN bit was set in the RXMBPENABLE.

**27.2.6.5.21 No Match (NOMATCH) Flag**

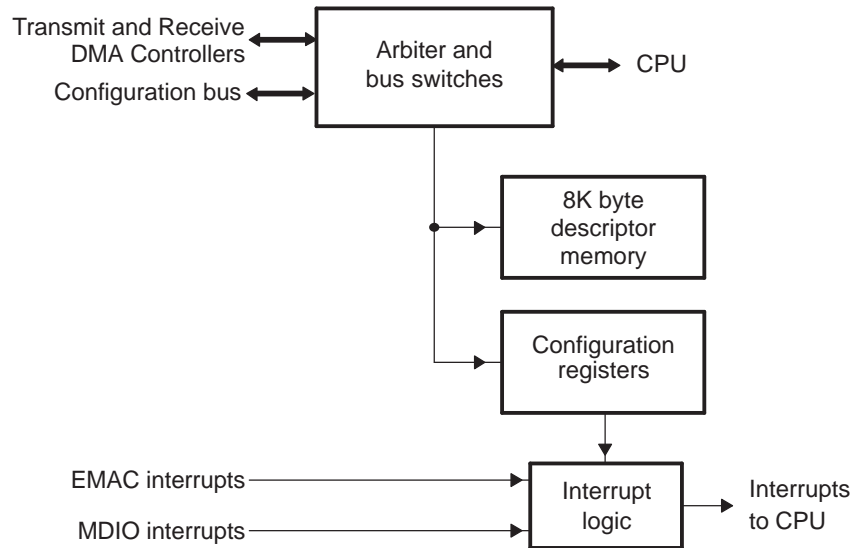
This flag is set by the EMAC in the SOP buffer descriptor, if the received packet did not pass any of the EMAC's address match criteria and was not discarded because the RXCAFEN bit was set in the RXMBPENABLE. Although the packet is a valid Ethernet data packet, it was only received because the EMAC is in promiscuous mode.



### 27.2.7 EMAC Control Module

The EMAC control module (Figure 27-9) interfaces the EMAC and MDIO modules to the rest of the system, and also provides a local memory space to hold EMAC packet buffer descriptors. Local memory is used to help avoid contention with device memory spaces. Other functions include the bus arbiter, and interrupt logic control.

Figure 27-9. EMAC Control Module Block Diagram



#### 27.2.7.1 Internal Memory

The EMAC control module includes 8K bytes of internal memory (CPPI buffer descriptor memory). The internal memory block is essential for allowing the EMAC to operate more independently of the CPU. It also prevents memory underflow conditions when the EMAC issues read or write requests to descriptor memory. (Memory accesses to read or write the actual Ethernet packet data are protected by the EMAC's internal FIFOs).

A descriptor is a 16-byte memory structure that holds information about a single Ethernet packet buffer, which may contain a full or partial Ethernet packet. Thus with the 8K memory block provided for descriptor storage, the EMAC module can send and received up to a combined 512 packets before it needs to be serviced by application or driver software.

#### 27.2.7.2 Bus Arbiter

The EMAC control module bus arbiter operates transparently to the rest of the system. It is used:

- To arbitrate between the CPU and EMAC buses for access to internal descriptor memory.
- To arbitrate between internal EMAC buses for access to system memory.



### 27.2.7.3 Interrupt Control

Interrupt conditions generated by the EMAC and MDIO modules are combined into four interrupt signals that are routed to the Vectored Interrupt Manager (VIM); the VIM then relays the interrupt signals to the CPU. The EMAC control module uses two sets of registers to control the interrupt signals to the CPU:

- C0RXTHRESHEN, C0RXEN, C0TXEN, and C0MISCEN registers enable the pulse signals that are mapped to the VIM
- INTCONTROL, C0RXIMAX, and C0TXIMAX registers enable interrupt pacing to limit the number of interrupt pulses generated per millisecond

Interrupts must be acknowledged by writing the appropriate value to the EMAC End-Of-Interrupt Vector (MACEOIVECTOR). The MACEOIVECTOR behaves as an interrupt pulse interlock -- once the EMAC control module has issued an interrupt pulse to the CPU, it will not generate further pulses of the same type until the original pulse has been acknowledged.

### 27.2.8 MDIO Module

The MDIO module is used to manage up to 32 physical layer (PHY) devices connected to the Ethernet Media Access Controller (EMAC). The device supports a single PHY being connected to the EMAC at any given time. The MDIO module is designed to allow almost transparent operation of the MDIO interface with little maintenance from the CPU.

The MDIO module continuously polls 32 MDIO addresses in order to enumerate all PHY devices in the system. Once a PHY device has been detected, the MDIO module reads the MDIO PHY link status register (LINK) to monitor the PHY link state. Link change events are stored in the MDIO module, which can interrupt the CPU. This storing of the events allows the CPU to poll the link status of the PHY device without continuously performing MDIO module accesses. However, when the CPU must access the MDIO module for configuration and negotiation, the MDIO module performs the MDIO read or write operation independent of the CPU. This independent operation allows the processor to poll for completion or interrupt the CPU once the operation has completed.

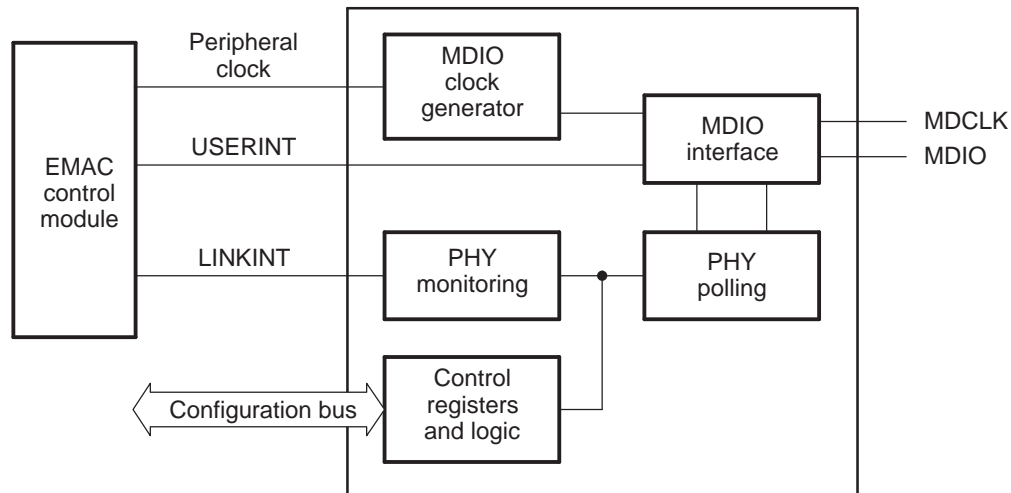
The MDIO module does not support the "Clause 45" interface.

#### 27.2.8.1 MDIO Module Components

The MDIO module ([Figure 27-10](#)) interfaces to the PHY components through two MDIO pins (MDIO\_CLK and MDIO), and to the CPU through the EMAC control module and the configuration bus. The MDIO module consists of the following logical components:

- MDIO clock generator
- Global PHY detection and link state monitoring
- Active PHY monitoring
- PHY register user access

**Figure 27-10. MDIO Module Block Diagram**



### 27.2.8.1.1 MDIO Clock Generator

The MDIO clock generator controls the MDIO clock based on a divide-down of the VCLK3 peripheral clock in the EMAC control module. The MDIO clock is specified to run up to 2.5 MHz, although typical operation would be 1.0 MHz. Since the VCLK3 peripheral clock frequency is configurable, the application software or driver controls the divide-down amount. See the device datasheet for peripheral clock speed specifications.

### 27.2.8.1.2 Global PHY Detection and Link State Monitoring

The MDIO module continuously polls all 32 MDIO addresses in order to enumerate the PHY devices in the system. The module tracks whether or not a PHY on a particular address has responded, and whether or not the PHY currently has a link. Using this information allows the software application to quickly determine which MDIO address the PHY is using.

### 27.2.8.1.3 Active PHY Monitoring

Once a PHY candidate has been selected for use, the MDIO module transparently monitors its link state by reading the MDIO PHY link status register (LINK). Link change events are stored on the MDIO device and can optionally interrupt the CPU. This allows the system to poll the link status of the PHY device without continuously performing costly MDIO accesses.

### 27.2.8.1.4 PHY Register User Access

When the CPU must access MDIO for configuration and negotiation, the PHY access module performs the actual MDIO read or write operation independent of the CPU. This allows the CPU to poll for completion or receive an interrupt when the read or write operation has been performed. The user access registers USERACCESS<sub>n</sub> allows the software to submit the access requests for the PHY connected to the device.

### 27.2.8.2 MDIO Module Operational Overview

The MDIO module implements the 802.3 serial management interface to interrogate and control an Ethernet PHY, using a shared two-wired bus. It separately performs autodetection and records the current link status of up to 32 PHYs, polling all 32 MDIO addresses.

Application software uses the MDIO module to configure the autonegotiation parameters of the PHY attached to the EMAC, retrieve the negotiation results, and configure required parameters in the EMAC.

In this device, the Ethernet PHY attached to the system can be directly controlled and queried. The Media Independent Interface (MII) address of this PHY device is specified in one of the PHYADRMON bits in the MDIO user PHY select register (USERPHYSEL $n$ ). The MDIO module can be programmed to trigger a CPU interrupt on a PHY link change event, by setting the LINKINTENB bit in USERPHYSEL $n$ . Reads and writes to registers in this PHY device are performed using the MDIO user access register (USERACCESS $n$ ).

The MDIO module powers-up in an idle state until specifically enabled by setting the ENABLE bit in the MDIO control register (CONTROL). At this time, the MDIO clock divider and preamble mode selection are also configured. The MDIO preamble is enabled by default, but can be disabled when the connected PHY does not require it. Once the MDIO module is enabled, the MDIO interface state machine continuously polls the PHY link status (by reading the generic status register) of all possible 32 PHY addresses and records the results in the MDIO PHY alive status register (ALIVE) and MDIO PHY link status register (LINK). The corresponding bit for the connected PHY (0-31) is set in ALIVE, if the PHY responded to the read request. The corresponding bit is set in LINK, if the PHY responded and also is currently linked. In addition, any PHY register read transactions initiated by the application software using USERACCESS $n$  causes ALIVE to be updated.

The USERPHYSEL $n$  is used to track the link status of the connected PHY address. A change in the link status of the PHY being monitored sets the appropriate bit in the MDIO link status change interrupt registers (LINKINTRAW and LINKINTMASKED), if enabled by the LINKINTENB bit in USERPHYSEL $n$ .

While the MDIO module is enabled, the host issues a read or write transaction over the MII management interface using the DATA, PHYADR, REGADR, and WRITE bits in USERACCESS $n$ . When the application sets the GO bit in USERACCESS $n$ , the MDIO module begins the transaction without any further intervention from the CPU. Upon completion, the MDIO module clears the GO bit and sets the corresponding USERINTRAW bit (0 or 1) in the MDIO user command complete interrupt register (USERINTRAW) corresponding to USERACCESS $n$  used. The corresponding USERINTMASKED bit (0 or 1) in the MDIO user command complete interrupt register (USERINTMASKED) may also be set, depending on the mask setting configured in the MDIO user command complete interrupt mask set register (USERINTMASKSET) and the MDIO user interrupt mask clear register (USERINTMASKCLEAR).

A round-robin arbitration scheme is used to schedule transactions that may be queued using both USERACCESS0 and USERACCESS1. The application software must check the status of the GO bit in USERACCESS $n$  before initiating a new transaction, to ensure that the previous transaction has completed. The application software can use the ACK bit in USERACCESS $n$  to determine the status of a read transaction.

### **27.2.8.2.1 Initializing the MDIO Module**

The following steps are performed by the application software or device driver to initialize the MDIO device:

1. Configure the PREAMBLE and CLKDIV bits in the MDIO control register (CONTROL).
2. Enable the MDIO module by setting the ENABLE bit in CONTROL.
3. The MDIO PHY alive status register (ALIVE) can be read in polling fashion until a PHY connected to the system responded, and the MDIO PHY link status register (LINK) can determine whether this PHY already has a link.
4. Setup the appropriate PHY addresses in the MDIO user PHY select register (USERPHYSEL $n$ ), and set the LINKINTENB bit to enable a link change event interrupt if desirable.
5. If an interrupt on general MDIO register access is desired, set the corresponding bit in the MDIO user command complete interrupt mask set register (USERINTMASKSET) to use the MDIO user access register (USERACCESS $n$ ). Since only one PHY is used in this device, the application software can use one USERACCESS $n$  to trigger a completion interrupt; the other USERACCESS $n$  is not setup.

### **27.2.8.2.2 Writing Data To a PHY Register**

The MDIO module includes a user access register (USERACCESS $n$ ) to directly access a specified PHY device. To write a PHY register, perform the following:

1. Check to ensure that the GO bit in the MDIO user access register (USERACCESS $n$ ) is cleared.
2. Write to the GO, WRITE, REGADR, PHYADR, and DATA bits in USERACCESS $n$  corresponding to the PHY and PHY register you want to write.
3. The write operation to the PHY is scheduled and completed by the MDIO module. Completion of the write operation can be determined by polling the GO bit in USERACCESS $n$  for a 0.
4. Completion of the operation sets the corresponding USERINTRAW bit (0 or 1) in the MDIO user command complete interrupt register (USERINTRAW) corresponding to USERACCESS $n$  used. If interrupts have been enabled on this bit using the MDIO user command complete interrupt mask set register (USERINTMASKSET), then the bit is also set in the MDIO user command complete interrupt register (USERINTMASKED) and an interrupt is triggered on the CPU.

### **27.2.8.2.3 Reading Data From a PHY Register**

The MDIO module includes a user access register (USERACCESS $n$ ) to directly access a specified PHY device. To read a PHY register, perform the following:

1. Check to ensure that the GO bit in the MDIO user access register (USERACCESS $n$ ) is cleared.
2. Write to the GO, REGADR, and PHYADR bits in USERACCESS $n$  corresponding to the PHY and PHY register you want to read.
3. The read data value is available in the DATA bits in USERACCESS $n$  after the module completes the read operation on the serial bus. Completion of the read operation can be determined by polling the GO and ACK bits in USERACCESS $n$ . Once the GO bit has cleared, the ACK bit is set on a successful read.
4. Completion of the operation sets the corresponding USERINTRAW bit (0 or 1) in the MDIO user command complete interrupt register (USERINTRAW) corresponding to USERACCESS $n$  used. If interrupts have been enabled on this bit using the MDIO user command complete interrupt mask set register (USERINTMASKSET), then the bit is also set in the MDIO user command complete interrupt register (USERINTMASKED) and an interrupt is triggered on the CPU.

### 27.2.8.2.4 Example of MDIO Register Access Code

The MDIO module uses the MDIO user access register (USERACCESS $n$ ) to access the PHY control registers. Software functions that implement the access process may simply be the following four macros:

- PHYREG\_read( regadr, phyadr )                      Start the process of reading a PHY register
- PHYREG\_write( regadr, phyadr, data )              Start the process of writing a PHY register
- PHYREG\_wait( )                                        Synchronize operation (make sure read/write is idle)
- PHYREG\_waitResults( results )                      Wait for read to complete and return data read

Note that it is not necessary to wait after a write operation, as long as the status is checked before every operation to make sure the MDIO hardware is idle. An alternative approach is to call PHYREG\_wait() after every write, and PHYREG\_waitResults( ) after every read, then the hardware can be assumed to be idle when starting a new operation.

The implementation of these macros using the chip support library (CSL) is shown in [Example 27-3](#) (USERACCESS0 is assumed).

Note that this implementation does not check the ACK bit in USERACCESS $n$  on PHY register reads (does not follow the procedure outlined in [Section 27.2.8.2.3](#)). Since the MDIO PHY alive status register (ALIVE) is used to initially select a PHY, it is assumed that the PHY is acknowledging read operations. It is possible that a PHY could become inactive at a future point in time. An example of this would be a PHY that can have its MDIO addresses changed while the system is running. It is not very likely, but this condition can be tested by periodically checking the PHY state in ALIVE.

#### Example 27-3. MDIO Register Access Macros

```
#define PHYREG_read(regadr, phyadr)
    MDIO_REGS->USERACCESS0 =
        CSL_FMK(MDIO_USERACCESS0_GO,1u)           | /
        CSL_FMK(MDIO_USERACCESS0_REGADR,regadr)   | /
        CSL_FMK(MDIO_USERACCESS0_PHYADR,phyadr)
#define PHYREG_write(regadr, phyadr, data)
    MDIO_REGS->USERACCESS0 =
        CSL_FMK(MDIO_USERACCESS0_GO,1u)           | /
        CSL_FMK(MDIO_USERACCESS0_WRITE,1)         | /
        CSL_FMK(MDIO_USERACCESS0_REGADR,regadr)   | /
        CSL_FMK(MDIO_USERACCESS0_PHYADR,phyadr)   | /
        CSL_FMK(MDIO_USERACCESS0_DATA, data)
#define PHYREG_wait()
    while( CSL_FEXT(MDIO_REGS->USERACCESS0,MDIO_USERACCESS0_GO) )
#define PHYREG_waitResults( results ) {
    while( CSL_FEXT(MDIO_REGS->USERACCESS0,MDIO_USERACCESS0_GO) );
    results = CSL_FEXT(MDIO_REGS->USERACCESS0, MDIO_USERACCESS0_DATA); }
```

## 27.2.9 EMAC Module

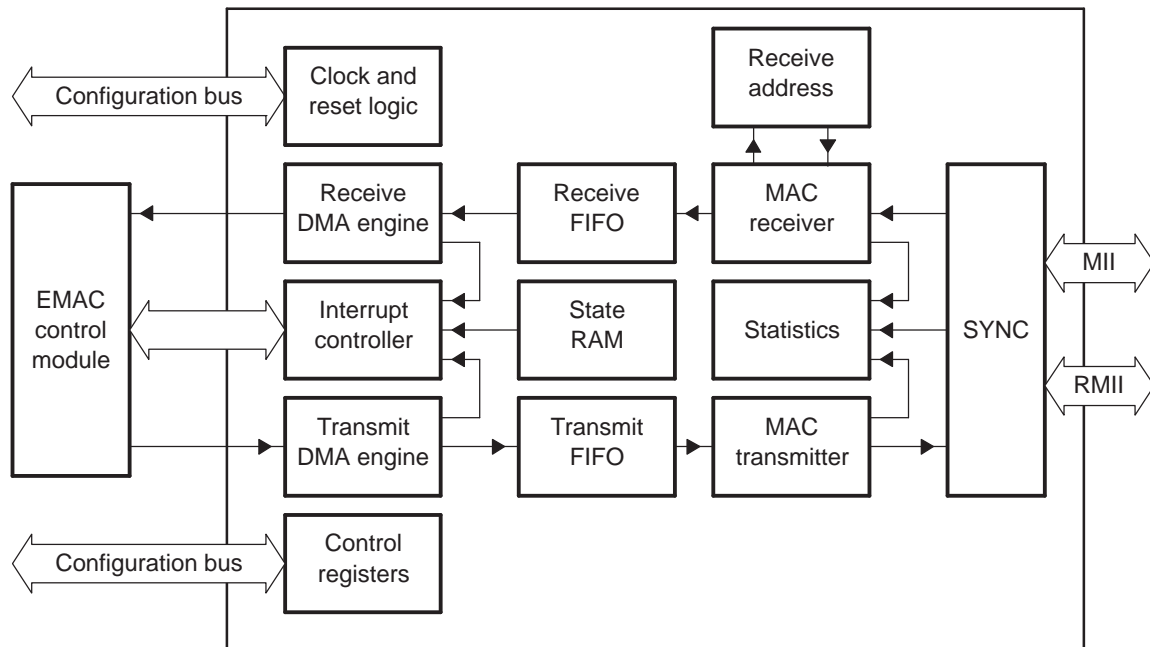
This section discusses the architecture and basic function of the EMAC module.

### 27.2.9.1 EMAC Module Components

The EMAC module (Figure 27-11) interfaces to the outside world through the Media Independent Interface (MII) or Reduced Media Independent Interface (RMII). The interface between the EMAC module and the system core is provided through the EMAC control module. The EMAC consists of the following logical components:

- The receive path includes: receive DMA engine, receive FIFO, and MAC receiver
- The transmit path includes: transmit DMA engine, transmit FIFO, and MAC transmitter
- Statistics logic
- State RAM
- Interrupt controller
- Control registers and logic
- Clock and reset logic

Figure 27-11. EMAC Module Block Diagram



#### 27.2.9.1.1 Receive DMA Engine

The receive DMA engine is the interface between the receive FIFO and the system core. It interfaces to the CPU through the bus arbiter in the EMAC control module. This DMA engine is totally independent of the device DMA.

#### 27.2.9.1.2 Receive FIFO

The receive FIFO consists of three cells of 64-bytes each and associated control logic. The FIFO buffers receive data in preparation for writing into packet buffers in device memory.

### 27.2.9.1.3 MAC Receiver

The MAC receiver detects and processes incoming network frames, de-frames them, and puts them into the receive FIFO. The MAC receiver also detects errors and passes statistics to the statistics RAM.

### 27.2.9.1.4 Transmit DMA Engine

The transmit DMA engine is the interface between the transmit FIFO and the CPU. It interfaces to the CPU through the bus arbiter in the EMAC control module.

### 27.2.9.1.5 Transmit FIFO

The transmit FIFO consists of three cells of 64-bytes each and associated control logic. The FIFO buffers data in preparation for transmission.

### 27.2.9.1.6 MAC Transmitter

The MAC transmitter formats frame data from the transmit FIFO and transmits the data using the CSMA/CD access protocol. The frame CRC can be automatically appended, if required. The MAC transmitter also detects transmission errors and passes statistics to the statistics registers.

### 27.2.9.1.7 Statistics Logic

The Ethernet statistics are counted and stored in the statistics logic RAM. This statistics RAM keeps track of 36 different Ethernet packet statistics.

### 27.2.9.1.8 State RAM

State RAM contains the head descriptor pointers and completion pointers registers for both transmit and receive channels.

### 27.2.9.1.9 EMAC Interrupt Controller

The interrupt controller contains the interrupt related registers and logic. The 26 raw EMAC interrupts are input to this submodule and masked module interrupts are output.

### 27.2.9.1.10 Control Registers and Logic

The EMAC is controlled by a set of memory-mapped registers. The control logic also signals transmit, receive, and status related interrupts to the CPU through the EMAC control module.

### 27.2.9.1.11 Clock and Reset Logic

The clock and reset submodule generates all the EMAC clocks and resets. For more details on reset capabilities, see [Section 27.2.15.1](#).

## 27.2.9.2 EMAC Module Operational Overview

After reset, initialization, and configuration, the host may initiate transmit operations. Transmit operations are initiated by host writes to the appropriate transmit channel head descriptor pointer contained in the state RAM block. The transmit DMA controller then fetches the first packet in the packet chain from memory. The DMA controller writes the packet into the transmit FIFO in bursts of 64-byte cells. When the threshold number of cells, configurable using the TXCELLTHRESH bit in the FIFO control register (FIFOCONTROL), have been written to the transmit FIFO, or a complete packet, whichever is smaller, the MAC transmitter then initiates the packet transmission. The SYNC block transmits the packet over the MII or RMII interfaces in accordance with the 802.3 protocol. Transmit statistics are counted by the statistics block.

Receive operations are initiated by host writes to the appropriate receive channel head descriptor pointer



after host initialization and configuration. The SYNC submodule receives packets and strips off the Ethernet related protocol. The packet data is input to the MAC receiver, which checks for address match and processes errors. Accepted packets are then written to the receive FIFO in bursts of 64-byte cells. The receive DMA controller then writes the packet data to memory. Receive statistics are counted by the statistics block.

The EMAC module operates independently of the CPU. It is configured and controlled by its register set mapped into device memory. Information about data packets is communicated by use of 16-byte descriptors that are placed in an 8K-byte block of RAM in the EMAC control module (CPPI buffer descriptor memory).

For transmit operations, each 16-byte descriptor describes a packet or packet fragment in the system's internal or external memory. For receive operations, each 16-byte descriptor represents a free packet buffer or buffer fragment. On both transmit and receive, an Ethernet packet is allowed to span one or more memory fragments, represented by one 16-byte descriptor per fragment. In typical operation, there is only one descriptor per receive buffer, but transmit packets may be fragmented, depending on the software architecture.

An interrupt is issued to the CPU whenever a transmit or receive operation has completed. However, it is not necessary for the CPU to service the interrupt while there are additional resources available. In other words, the EMAC continues to receive Ethernet packets until its receive descriptor list has been exhausted. On transmit operations, the transmit descriptors need only be serviced to recover their associated memory buffer. Thus, it is possible to delay servicing of the EMAC interrupt if there are real-time tasks to perform.

Eight channels are supplied for both transmit and receive operations. On transmit, the eight channels represent eight independent transmit queues. The EMAC can be configured to treat these channels as an equal priority "round-robin" queue or as a set of eight fixed-priority queues. On receive, the eight channels represent eight independent receive queues with packet classification. Packets are classified based on the destination MAC address. Each of the eight channels is assigned its own MAC address, enabling the EMAC module to act like eight virtual MAC adapters. Also, specific types of frames can be sent to specific channels. For example, multicast, broadcast, or other (promiscuous, error, etc.), can each be received on a specific receive channel queue.

The EMAC keeps track of 36 different statistics, plus keeps the status of each individual packet in its corresponding packet descriptor.

## **27.2.10 MAC Interface**

The following sections discuss the operation of the Media Independent Interface (MII) and Reduced Media Independent Interface (RMII) in 10 Mbps and 100 Mbps mode. An IEEE 802.3 compliant Ethernet MAC controls the interface.

### **27.2.10.1 Data Reception**

#### **27.2.10.1.1 Receive Control**

Data received from the PHY is interpreted and output to the EMAC receive FIFO. Interpretation involves detection and removal of the preamble and start-of-frame delimiter, extraction of the address and frame length, data handling, error checking and reporting, cyclic redundancy checking (CRC), and statistics control signal generation. Address detection and frame filtering is performed outside the MAC interface.

#### **27.2.10.1.2 Receive Inter-Frame Interval**

The 802.3 standard requires an interpacket gap (IPG), which is 96 bit times. However, the EMAC can tolerate a reduced IPG of 8 bit times with a correct preamble and start frame delimiter. This interval between frames must comprise (in the following order):

1. An Interpacket Gap (IPG).
2. A 7-byte preamble (all bytes 55h).
3. A 1-byte start of frame delimiter (5Dh).



### 27.2.10.1.3 Receive Flow Control

When enabled and triggered, receive flow control is initiated to limit the EMAC from further frame reception. Two forms of receive buffer flow control are available:

- Collision-based flow control for half-duplex mode
- IEEE 802.3x pause frames flow control for full-duplex mode

In either case, receive flow control prevents frame reception by issuing the flow control appropriate for the current mode of operation. Receive flow control prevents reception of frames on the EMAC until all of the triggering conditions clear, at which time frames may again be received by the EMAC.

Receive flow control is enabled by the RXBUFFERFLOWEN bit in the MAC control register (MACCONTROL). The EMAC is configured for collision or IEEE 802.3X flow control using the FULLDUPLEX bit in MACCONTROL. Receive flow control is triggered when the number of free buffers in any enabled receive channel free buffer count register (RXnFREEBUFFER) is less than or equal to the receive channel flow control threshold register (RXnFLOWTHRESH) value. Receive flow control is independent of receive QOS, except that both use the free buffer values.

#### 27.2.10.1.3.1 Collision-Based Receive Buffer Flow Control

Collision-based receive buffer flow control provides a means of preventing frame reception when the EMAC is operating in half-duplex mode (the FULLDUPLEX bit is cleared in MACCONTROL). When receive flow control is enabled and triggered, the EMAC generates collisions for received frames. The jam sequence transmitted is the 12-byte sequence C3.C3.C3.C3.C3.C3.C3.C3.C3.C3.C3.C3h. The jam sequence begins no later than approximately as the source address starts to be received. Note that these forced collisions are not limited to a maximum of 16 consecutive collisions, and are independent of the normal back-off algorithm.

Receive flow control does not depend on the value of the incoming frame destination address. A collision is generated for any incoming packet, regardless of the destination address, if any EMAC enabled channel's free buffer register value is less than or equal to the channel's flow threshold value.

#### 27.2.10.1.3.2 IEEE 802.3x-Based Receive Buffer Flow Control

IEEE 802.3x-based receive buffer flow control provides a means of preventing frame reception when the EMAC is operating in full-duplex mode (the FULLDUPLEX bit is set in MACCONTROL). When receive flow control is enabled and triggered, the EMAC transmits a pause frame to request that the sending station stop transmitting for the period indicated within the transmitted pause frame.

The EMAC transmits a pause frame to the reserved multicast address at the first available opportunity (immediately if currently idle or following the completion of the frame currently being transmitted). The pause frame contains the maximum possible value for the pause time (FFFFh). The EMAC counts the receive pause frame time (decrements FF00h to 0) and retransmits an outgoing pause frame, if the count reaches 0. When the flow control request is removed, the EMAC transmits a pause frame with a zero pause time to cancel the pause request.

Note that transmitted pause frames are only a request to the other end station to stop transmitting. Frames that are received during the pause interval are received normally (provided the receive FIFO is not full).

Pause frames are transmitted if enabled and triggered, regardless of whether or not the EMAC is observing the pause time period from an incoming pause frame.

The EMAC transmits pause frames as described below:

- The 48-bit reserved multicast destination address 01.80.C2.00.00.01h.
- The 48-bit source address (set using the MACSRCADDRLO and MACSRCADDRHI registers).
- The 16-bit length/type field containing the value 88.08h.
- The 16-bit pause opcode equal to 00.01h.
- The 16-bit pause time value of FF.FFh. A pause-quantum is 512 bit-times. Pause frames sent to cancel a pause request have a pause time value of 00.00h.
- Zero padding to 64-byte data length (EMAC transmits only 64-byte pause frames).

- The 32-bit frame-check sequence (CRC word).

All quantities are hexadecimal and are transmitted most-significant byte first. The least-significant bit (LSB) is transferred first in each byte.

If the RXBUFFERFLOWEN bit in MACCONTROL is cleared to 0 while the pause time is nonzero, then the pause time is cleared to 0 and a zero count pause frame is sent.

### **27.2.10.2 Data Transmission**

The EMAC passes data to the PHY from the transmit FIFO (when enabled). Data is synchronized to the transmit clock rate. Transmission begins when there are TXCELLTHRESH cells of 64 bytes each, or a complete packet, in the FIFO.

#### **27.2.10.2.1 Transmit Control**

A jam sequence is output if a collision is detected on a transmit packet. If the collision was late (after the first 64 bytes have been transmitted), the collision is ignored. If the collision is not late, the controller will back off before retrying the frame transmission. When operating in full-duplex mode, the carrier sense (MII\_CRIS) and collision-sensing (MII\_COL) modes are disabled.

#### **27.2.10.2.2 CRC Insertion**

If the SOP buffer descriptor PASSCRC flag is cleared, the EMAC generates and appends a 32-bit Ethernet CRC onto the transmitted data. For the EMAC-generated CRC case, a CRC (or placeholder) at the end of the data is allowed but not required. The buffer byte count value should not include the CRC bytes, if they are present.

If the SOP buffer descriptor PASSCRC flag is set, then the last four bytes of the transmit data are transmitted as the frame CRC. The four CRC data bytes should be the last four bytes of the frame and should be included in the buffer byte count value. The MAC performs no error checking on the outgoing CRC.

#### **27.2.10.2.3 Adaptive Performance Optimization (APO)**

The EMAC incorporates adaptive performance optimization (APO) logic that may be enabled by setting the TXPACE bit in the MAC control register (MACCONTROL). Transmission pacing to enhance performance is enabled when the TXPACE bit is set. Adaptive performance pacing introduces delays into the normal transmission of frames, delaying transmission attempts between stations, reducing the probability of collisions occurring during heavy traffic (as indicated by frame deferrals and collisions), thereby, increasing the chance of successful transmission.

When a frame is deferred, suffers a single collision, multiple collisions, or excessive collisions, the pacing counter is loaded with an initial value of 31. When a frame is transmitted successfully (without experiencing a deferral, single collision, multiple collision, or excessive collision), the pacing counter is decremented by 1, down to 0.

With pacing enabled, a new frame is permitted to immediately (after one interpacket gap) attempt transmission only if the pacing counter is 0. If the pacing counter is nonzero, the frame is delayed by the pacing delay of approximately four interpacket gap (IPG) delays. APO only affects the IPG preceding the first attempt at transmitting a frame; APO does not affect the back-off algorithm for retransmitted frames.

#### **27.2.10.2.4 Interpacket-Gap (IPG) Enforcement**

The measurement reference for the IPG of 96 bit times is changed depending on frame traffic conditions. If a frame is successfully transmitted without collision and MII\_CRIS is deasserted within approximately 48 bit times of MII\_TXEN being deasserted, then 96 bit times is measured from MII\_TXEN. If the frame suffered a collision or MII\_CRIS is not deasserted until more than approximately 48 bit times after MII\_TXEN is deasserted, then 96 bit times (approximately, but not less) is measured from MII\_CRIS.

#### **27.2.10.2.5 Back Off**

The EMAC implements the 802.3 binary exponential back-off algorithm.

### 27.2.10.2.6 Transmit Flow Control

Incoming pause frames are acted upon, when enabled, to prevent the EMAC from transmitting any further frames. Incoming pause frames are only acted upon when the FULLDUPLEX and TXFLOWEN bits in the MAC control register (MACCONTROL) are set. Pause frames are not acted upon in half-duplex mode. Pause frame action is taken if enabled, but normally the frame is filtered and not transferred to memory. MAC control frames are transferred to memory, if the RXCMFEN bit in the receive multicast/broadcast/promiscuous channel enable register (RXMBPENABLE) is set. The TXFLOWEN and FULLDUPLEX bits affect whether or not MAC control frames are acted upon, but they have no affect upon whether or not MAC control frames are transferred to memory or filtered.

Pause frames are a subset of MAC control frames with an opcode field of 0001h. Incoming pause frames are only acted upon by the EMAC if:

- TXFLOWEN bit is set in MACCONTROL
- The frame's length is 64 to RXMAXLEN bytes inclusive
- The frame contains no CRC error or align/code errors

The pause time value from valid frames is extracted from the two bytes following the opcode. The pause time is loaded into the EMAC transmit pause timer and the transmit pause time period begins. If a valid pause frame is received during the transmit pause time period of a previous transmit pause frame then:

- If the destination address is not equal to the reserved multicast address or any enabled or disabled unicast address, then the transmit pause timer immediately expires, or
- If the new pause time value is 0, then the transmit pause timer immediately expires, else
- The EMAC transmit pause timer immediately is set to the new pause frame pause time value. (Any remaining pause time from the previous pause frame is discarded).

If the TXFLOWEN bit in MACCONTROL is cleared, then the pause timer immediately expires.

The EMAC does not start the transmission of a new data frame any sooner than 512 bit-times after a pause frame with a nonzero pause time has finished being received (MII\_RXDV going inactive). No transmission begins until the pause timer has expired (the EMAC may transmit pause frames in order to initiate outgoing flow control). Any frame already in transmission when a pause frame is received is completed and unaffected.

Incoming pause frames consist of:

- A 48-bit destination address equal to one of the following:
  - The reserved multicast destination address 01.80.C2.00.00.01h
  - Any EMAC 48-bit unicast address. Pause frames are accepted, regardless of whether the channel is enabled or not.
- The 16-bit length/type field containing the value 88.08h.
- The 48-bit source address of the transmitting device.
- The 16-bit pause opcode equal to 00.01h.
- The 16-bit pause time. A pause-quantum is 512 bit-times.
- Padding to 64-byte data length.
- The 32-bit frame-check sequence (CRC word).

All quantities are hexadecimal and are transmitted most-significant byte first. The least-significant bit (LSB) is transferred first in each byte.

The padding is required to make up the frame to a minimum of 64 bytes. The standard allows pause frames longer than 64 bytes to be discarded or interpreted as valid pause frames. The EMAC recognizes any pause frame between 64 bytes and RXMAXLEN bytes in length.

### 27.2.10.2.7 Speed, Duplex, and Pause Frame Support

The MAC operates at 10 Mbps or 100 Mbps, in half-duplex or full-duplex mode, and with or without pause frame support as configured by the host.

## 27.2.11 Packet Receive Operation

### 27.2.11.1 Receive DMA Host Configuration

To configure the receive DMA for operation the host must:

- Initialize the receive addresses.
- Initialize the receive channel  $n$  DMA head descriptor pointer registers (RX $n$ HDP) to 0.
- Write the MAC address hash  $n$  registers (MACHASH1 and MACHASH2), if multicast addressing is desired.
- If flow control is to be enabled, initialize:
  - the receive channel  $n$  free buffer count registers (RX $n$ FREEBUFFER)
  - the receive channel  $n$  flow control threshold register (RX $n$ FLOWTHRESH)
  - the receive filter low priority frame threshold register (RXFILTERLOWTHRESH)
- Enable the desired receive interrupts using the receive interrupt mask set register (RXINTMASKSET) and the receive interrupt mask clear register (RXINTMASKCLEAR).
- Set the appropriate configuration bits in the MAC control register (MACCONTROL).
- Write the receive buffer offset register (RXBUFFEROFFSET) value (typically zero).
- Setup the receive channel(s) buffer descriptors and initialize RX $n$ HDP.
- Enable the receive DMA controller by setting the RXEN bit in the receive control register (RXCONTROL).
- Configure and enable the receive operation, as desired, in the receive multicast/broadcast/promiscuous channel enable register (RXMBPENABLE) and by using the receive unicast set register (RXUNICASTSET) and the receive unicast clear register (RXUNICASTCLEAR).

### 27.2.11.2 Receive Channel Enabling

Each of the eight receive channels has an enable bit (RXCH $n$ EN) in the receive unicast set register (RXUNICASTSET) that is controlled using RXUNICASTSET and the receive unicast clear register (RXUNICASTCLEAR). The RXCH $n$ EN bits determine whether the given channel is enabled (when set to 1) to receive frames with a matching unicast or multicast destination address.

The RXBROADEN bit in the receive multicast/broadcast/promiscuous channel enable register (RXMBPENABLE) determines if broadcast frames are enabled or filtered. If broadcast frames are enabled (when set to 1), then they are copied to only a single channel selected by the RXBROADCH bit in RXMBPENABLE.

The RXMULTEN bit in RXMBPENABLE determines if hash matching multicast frames are enabled or filtered. Incoming multicast addresses (group addresses) are hashed into an index in the hash table. If the indexed bit is set then the frame hash matches and will be transferred to the channel selected by the RXMULTCH bit in RXMBPENABLE when multicast frames are enabled. The multicast hash bits are set in the MAC address hash  $n$  registers (MACHASH1 and MACHASH2).

The RXPROMCH bit in RXMBPENABLE selects the promiscuous channel to receive frames selected by the RXCMFEN, RXCSFEN, RXCEFEN, and RXCAFEN bits. These four bits allow reception of MAC control frames, short frames, error frames, and all frames (promiscuous), respectively.

### 27.2.11.3 Receive Address Matching

All eight MAC addresses corresponding to the eight receive channels share the upper 40 bits. Only the lower byte is unique for each address. All eight receive addresses should be initialized, because pause frames are acted upon regardless of whether a channel is enabled or not.

A MAC address is written by first writing the address number (channel) to be written into the MAC index register (MACINDEX). The upper 32 bits of address are then written to the MAC address high bytes register (MACADDRHI), which is followed by writing the lower 16 bits of address to the MAC address low bytes register (MACADDRLO). Since all eight MAC addresses share the upper 40 bits of address, MACADDRHI needs to be written only the first time (for the first channel configured).

#### 27.2.11.4 Hardware Receive QOS Support

Hardware receive quality of service (QOS) is supported, when enabled, by the Tag Protocol Identifier format and the associated Tag Control Information (TCI) format priority field. When the incoming frame length/type value is equal to 81.00h, the EMAC recognizes the frame as an Ethernet Encoded Tag Protocol Type. The two octets immediately following the protocol type contain the 16-bit TCI field. Bits 15-13 of the TCI field contain the received frames priority (0 to 7). The received frame is a low-priority frame, if the priority value is 0 to 3; the received frame is a high-priority frame, if the priority value is 4 to 7. All frames that have a length/type field value not equal to 81.00h are low-priority frames. Received frames that contain priority information are determined by the EMAC as:

- A 48-bit (6 bytes) destination address equal to:
  - The destination station's individual unicast address.
  - The destination station's multicast address (MACHASH1 and MACHASH2).
  - The broadcast address of all ones.
- A 48-byte (6 bytes) source address.
- The 16-bit (2 bytes) length/type field containing the value 81.00h.
- The 16-bit (2 bytes) TCI field with the priority field in the upper 3 bits.
- Data bytes
- The 4 bytes CRC.

The receive filter low priority frame threshold register (RXFILTERLOWTHRESH) and the receive channel  $n$  free buffer count registers (RX $n$ FREEBUFFER) are used in conjunction with the priority information to implement receive hardware QOS. Low-priority frames are filtered if the number of free buffers (RX $n$ FREEBUFFER) for the frame channel is less than or equal to the filter low threshold (RXFILTERLOWTHRESH) value. Hardware QOS is enabled by the RXQOSEN bit in the receive multicast/broadcast/promiscuous channel enable register (RXMBPENABLE).

#### 27.2.11.5 Host Free Buffer Tracking

The host must track free buffers for each enabled channel (including unicast, multicast, broadcast, and promiscuous), if receive QOS or receive flow control is used. Disabled channel free buffer values are do not cares. During initialization, the host should write the number of free buffers for each enabled channel to the appropriate receive channel  $n$  free buffer count registers (RX $n$ FREEBUFFER). The EMAC decrements the appropriate channel's free buffer value for each buffer used. When the host reclaims the frame buffers, the host should write the channel free buffer register with the number of reclaimed buffers (write to increment). There are a maximum of 65,535 free buffers available. RX $n$ FREEBUFFER only needs to be updated by the host if receive QOS or flow control is used.

#### 27.2.11.6 Receive Channel Teardown

The host commands a receive channel teardown by writing the channel number to the receive teardown register (RXTEARDOWN). When a teardown command is issued to an enabled receive channel, the following occurs:

- Any current frame in reception completes normally.
- The TDOWNCMPLT flag is set in the next buffer descriptor in the chain, if there is one.
- The channel head descriptor pointer is cleared to 0.
- A receive interrupt for the channel is issued to the host.
- The corresponding receive channel  $n$  completion pointer register (RX $n$ CP) contains the value FFFF FFCh.

Channel teardown may be commanded on any channel at any time. The host is informed of the teardown completion by the set teardown complete (TDOWNCMPLT) buffer descriptor bit. The EMAC does not clear any channel enables due to a teardown command. A teardown command to an inactive channel issues an interrupt that software should acknowledge with an FFFF FFFCh acknowledge value to RX $n$ CP (note that there is no buffer descriptor in this case). Software may read RX $n$ CP to determine if the interrupt was due to a commanded teardown. The read value is FFFF FFFCh, if the interrupt was due to a teardown command.



### 27.2.11.7 Receive Frame Classification

Received frames are proper (good) frames, if they are between 64 bytes and the value in the receive maximum length register (RXMAXLEN) bytes in length (inclusive) and contain no code, align, or CRC errors.

Received frames are long frames, if their frame count exceeds the value in RXMAXLEN. The RXMAXLEN reset (default) value is 5EEh (1518 in decimal). Long received frames are either oversized or jabber frames. Long frames with no errors are oversized frames; long frames with CRC, code, or alignment errors are jabber frames.

Received frames are short frames, if their frame count is less than 64 bytes. Short frames that address match and contain no errors are undersized frames; short frames with CRC, code, or alignment errors are fragment frames. If the frame length is less than or equal to 20, then the frame CRC is passed, regardless of whether the RXPASSCRC bit is set or cleared in the receive multicast/broadcast/promiscuous channel enable register (RXMBPENABLE).

A received long packet always contains RXMAXLEN number of bytes transferred to memory (if the RXCEFEN bit is set in RXMBPENABLE), regardless of the value of the RXPASSCRC bit. Following is an example with RXMAXLEN set to 1518:

- If the frame length is 1518, then the packet is not a long packet and there are 1514 or 1518 bytes transferred to memory depending on the value of the RXPASSCRC bit.
- If the frame length is 1519, there are 1518 bytes transferred to memory regardless of the RXPASSCRC bit value. The last three bytes are the first three CRC bytes.
- If the frame length is 1520, there are 1518 bytes transferred to memory regardless of the RXPASSCRC bit value. The last two bytes are the first two CRC bytes.
- If the frame length is 1521, there are 1518 bytes transferred to memory regardless of the RXPASSCRC bit value. The last byte is the first CRC byte.
- If the frame length is 1522, there are 1518 bytes transferred to memory. The last byte is the last data byte.

### 27.2.11.8 Promiscuous Receive Mode

When the promiscuous receive mode is enabled by setting the RXCAFEN bit in the receive multicast/broadcast/promiscuous channel enable register (RXMBPENABLE), nonaddress matching frames that would normally be filtered are transferred to the promiscuous channel. Address matching frames that would normally be filtered due to errors are transferred to the address match channel when the RXCAFEN and RXCEFEN bits in RXMBPENABLE are set. A frame is considered to be an address matching frame only if it is enabled to be received on a unicast, multicast, or broadcast channel. Frames received to disabled unicast, multicast, or broadcast channels are considered nonaddress matching.

MAC control frames address match only if the RXCMFEN bit in RXMBPENABLE is set. The RXCEFEN and RXCSFEN bits in RXMBPENABLE determine whether error frames are transferred to memory or not, but they do not determine whether error frames are address matching or not. Short frames are a special type of error frames.

A single channel is selected as the promiscuous channel by the RXPROMCH bit in RXMBPENABLE. The promiscuous receive mode is enabled by the RXCMFEN, RXCEFEN, RXCSFEN, and RXCAFEN bits in RXMBPENABLE. [Table 27-7](#) shows the effects of the promiscuous enable bits. Proper frames are frames that are between 64 bytes and the value in the receive maximum length register (RXMAXLEN) bytes in length inclusive and contain no code, align, or CRC errors.

**Table 27-7. Receive Frame Treatment Summary**

Address Match	RXCAFEN	RXCEFEN	RXCMFEN	RXCSFEN	Receive Frame Treatment
0	0	X	X	X	No frames transferred.
0	1	0	0	0	Proper frames transferred to promiscuous channel.
0	1	0	0	1	Proper/undersized data frames transferred to promiscuous channel.
0	1	0	1	0	Proper data and control frames transferred to promiscuous channel.
0	1	0	1	1	Proper/undersized data and control frames transferred to promiscuous channel.
0	1	1	0	0	Proper/oversize/jabber/code/align/CRC data frames transferred to promiscuous channel. No control or undersized/fragment frames are transferred.
0	1	1	0	1	Proper/undersized/fragment/oversize/jabber/code/align/CRC data frames transferred to promiscuous channel. No control frames are transferred.
0	1	1	1	0	Proper/oversize/jabber/code/align/CRC data and control frames transferred to promiscuous channel. No undersized frames are transferred.
0	1	1	1	1	All nonaddress matching frames with and without errors transferred to promiscuous channel.
1	X	0	0	0	Proper data frames transferred to address match channel.
1	X	0	0	1	Proper/undersized data frames transferred to address match channel.
1	X	0	1	0	Proper data and control frames transferred to address match channel.
1	X	0	1	1	Proper/undersized data and control frames transferred to address match channel.
1	X	1	0	0	Proper/oversize/jabber/code/align/CRC data frames transferred to address match channel. No control or undersized frames are transferred.
1	X	1	0	1	Proper/oversize/jabber/fragment/undersized/code/align/CRC data frames transferred to address match channel. No control frames are transferred.
1	X	1	1	0	Proper/oversize/jabber/code/align/CRC data and control frames transferred to address match channel. No undersized/fragment frames are transferred.
1	X	1	1	1	All address matching frames with and without errors transferred to the address match channel

### 27.2.11.9 Receive Overrun

The types of receive overrun are:

- FIFO start of frame overrun (FIFO\_SOF)
- FIFO middle of frame overrun (FIFO\_MOF)
- DMA start of frame overrun (DMA\_SOF)
- DMA middle of frame overrun (DMA\_MOF)

The statistics counters used to track these types of receive overrun are:

- Receive start of frame overruns register (RXSOFOVERRUNS)
- Receive middle of frame overruns register (RXMOFOVERRUNS)
- Receive DMA overruns register (RXDMAOVERRUNS)

Start of frame overruns happen when there are no resources available when frame reception begins. Start of frame overruns increment the appropriate overrun statistic(s) and the frame is filtered.

Middle of frame overruns happen when there are some resources to start the frame reception, but the resources run out during frame reception. In normal operation, a frame that overruns after starting the frame reception is filtered and the appropriate statistic(s) are incremented; however, the RXCEFEN bit in the receive multicast/broadcast/promiscuous channel enable register (RXMBPENABLE) affects overrun frame treatment. [Table 27-8](#) shows how the overrun condition is handled for the middle of frame overrun.

**Table 27-8. Middle of Frame Overrun Treatment**

Address Match	RXCAFEN	RXCEFEN	Middle of Frame Overrun Treatment
0	0	X	Overrun frame filtered.
0	1	0	Overrun frame filtered.
0	1	1	As much frame data as possible is transferred to the promiscuous channel until overrun. The appropriate overrun statistic(s) is incremented and the OVERRUN and NOMATCH flags are set in the SOP buffer descriptor. Note that the RXMAXLEN number of bytes cannot be reached for an overrun to occur (it would be truncated and be a jabber or oversize).
1	X	0	Overrun frame filtered with the appropriate overrun statistic(s) incremented.
1	X	1	As much frame data as possible is transferred to the address match channel until overrun. The appropriate overrun statistic(s) is incremented and the OVERRUN flag is set in the SOP buffer descriptor. Note that the RXMAXLEN number of bytes cannot be reached for an overrun to occur (it would be truncated).



## 27.2.12 Packet Transmit Operation

The transmit DMA is an eight channel interface. Priority between the eight queues may be either fixed or round-robin as selected by the TXPTYPE bit in the MAC control register (MACCONTROL). If the priority type is fixed, then channel 7 has the highest priority and channel 0 has the lowest priority. Round-robin priority proceeds from channel 0 to channel 7.

### 27.2.12.1 Transmit DMA Host Configuration

To configure the transmit DMA for operation the host must perform:

- Write the MAC source address low bytes register (MACSRCADDRLO) and the MAC source address high bytes register (MACSRCADDRHI) (used for pause frames on transmit).
- Initialize the transmit channel  $n$  DMA head descriptor pointer registers (TX $n$ HDP) to 0.
- Enable the desired transmit interrupts using the transmit interrupt mask set register (TXINTMASKSET) and the transmit interrupt mask clear register (TXINTMASKCLEAR).
- Set the appropriate configuration bits in the MAC control register (MACCONTROL).
- Setup the transmit channel(s) buffer descriptors in host memory.
- Enable the transmit DMA controller by setting the TXEN bit in the transmit control register (TXCONTROL).
- Write the appropriate TX $n$ HDP with the pointer to the first descriptor to start transmit operations.

### 27.2.12.2 Transmit Channel Teardown

The host commands a transmit channel teardown by writing the channel number to the transmit teardown register (TXTEARDOWN). When a teardown command is issued to an enabled transmit channel, the following occurs:

- Any frame currently in transmission completes normally.
- The TDOWNCMPLT flag is set in the next SOP buffer descriptor in the chain, if there is one.
- The channel head descriptor pointer is cleared to 0.
- A transmit interrupt is issued to inform the host of the channel teardown.
- The corresponding transmit channel  $n$  completion pointer register (TX $n$ CP) contains the value FFFF FFFCh.
- The host should acknowledge a teardown interrupt with an FFFF FFFCh acknowledge value.

Channel teardown may be commanded on any channel at any time. The host is informed of the teardown completion by the set teardown complete (TDOWNCMPLT) buffer descriptor bit. The EMAC does not clear any channel enables due to a teardown command. A teardown command to an inactive channel issues an interrupt that software should acknowledge with an FFFF FFFCh acknowledge value to TX $n$ CP (note that there is no buffer descriptor in this case). Software may read the interrupt acknowledge location (TX $n$ CP) to determine if the interrupt was due to a commanded teardown. The read value is FFFF FFFCh, if the interrupt was due to a teardown command.

### **27.2.13 Receive and Transmit Latency**

The transmit and receive FIFOs each contain three 64-byte cells. The EMAC begins transmission of a packet on the wire after TXCELLTHRESH (configurable through the FIFO control register) cells, or a complete packet, are available in the FIFO.

Transmit underrun cannot occur for packet sizes of TXCELLTHRESH times 64 bytes (or less). For larger packet sizes, transmit underrun occurs if the memory latency is greater than the time required to transmit a 64-byte cell on the wire; this is 5.12  $\mu$ s in 100 Mbps mode and 51.2  $\mu$ s in 10 Mbps mode. The memory latency time includes all buffer descriptor reads for the entire cell data.

Receive overrun is prevented if the receive memory cell latency is less than the time required to transmit a 64-byte cell on the wire: 5.12  $\mu$ s in 100 Mbps mode, or 51.2  $\mu$ s in 10 Mbps mode. The latency time includes any required buffer descriptor reads for the cell data.

Latency to system's internal and external RAM can be controlled through the use of the transfer node priority allocation register available at the device level. Latency to descriptor RAM is low because RAM is local to the EMAC, as it is part of the EMAC control module.

### **27.2.14 Transfer Node Priority**

The device contains a chip-level master priority register that is used to set the priority of the transfer node used in issuing memory transfer requests to system memory.

Although the EMAC has internal FIFOs to help alleviate memory transfer arbitration problems, the average transfer rate of data read and written by the EMAC to internal or external processor memory must be at least that of the Ethernet wire rate. In addition, the internal FIFO system can not withstand a single memory latency event greater than the time it takes to fill or empty a TXCELLTHRESH number of internal 64 byte FIFO cells.

For 100 Mbps operation, these restrictions translate into the following rules:

- The short-term average, each 64-byte memory read/write request from the EMAC must be serviced in no more than 5.12  $\mu$ s.
- Any single latency event in request servicing can be no longer than  $(5.12 \times \text{TXCELLTHRESH}) \mu$ s.

## 27.2.15 Reset Considerations

### 27.2.15.1 Software Reset Considerations

The peripheral clock is controlled by the Global Clock Module (GCM), while the reset to the EMAC, MDIO and EMAC control module is controlled by the system module. See the "Architecture" chapter of the Technical Reference Manual for more on how to enable or disable the peripheral clock to the EMAC, MDIO and EMAC control module. For more on how the EMAC, MDIO, and EMAC control module are disabled or placed in reset at runtime, see [Section 27.2.18](#).

Within the peripheral itself, the EMAC component of the Ethernet MAC peripheral can be placed in a reset state by writing to the soft reset register (SOFTRESET). Writing a 1 to the SOFTRESET bit causes the EMAC logic to be reset and the register values to be set to their default values. Software reset occurs when the receive and transmit DMA controllers are in an idle state to avoid locking up the configuration bus; it is the responsibility of the software to verify that there are no pending frames to be transferred. After writing a 1 to the SOFTRESET bit, it may be polled to determine if the reset has occurred. If a 1 is read, the reset has not yet occurred; if a 0 is read, then a reset has occurred.

After a software reset operation, all the EMAC registers need to be reinitialized for proper data transmission, including the FULLDUPLEX bit setting in the MAC control register (MACCONTROL).

Unlike the EMAC module, the MDIO and EMAC control modules cannot be placed in reset from a register inside their memory map.

### 27.2.15.2 Hardware Reset Considerations

When a hardware reset occurs, the EMAC peripheral has its register values reset and all the components return to their default state. After the hardware reset, the EMAC needs to be initialized before being able to resume its data transmission, as described in [Section 27.2.16](#).

A hardware reset is the only means of recovering from the error interrupts (HOSTPEND), which are triggered by errors in packet buffer descriptors. Before doing a hardware reset, you should inspect the error codes in the MAC status register (MACSTATUS) that gives information about the type of software error that needs to be corrected. For detailed information on error interrupts, see [Section 27.2.17.1.4](#).

## **27.2.16 Initialization**

### **27.2.16.1 Enabling the EMAC/MDIO Peripheral**

When the device is powered on, the EMAC peripheral becomes enabled as soon as the system reset is released, and the EMAC peripheral registers are set to their default values. The application software can configure the EMAC peripheral registers as required.

### **27.2.16.2 EMAC Control Module Initialization**

The EMAC control module is used for global interrupt enables and to pace interrupts using 1 ms time windows. There is also an 8K block of CPPI RAM local to the EMAC that is used to hold packet buffer descriptors.

Note that although the EMAC control module and the EMAC module have slightly different functions, in practice, the type of maintenance performed on the EMAC control module is more commonly conducted from the EMAC module software (as opposed to the MDIO module).

The initialization of the EMAC control module consists of two parts:

1. Configuration of the interrupt to the CPU.
2. Initialization of the EMAC control module:
  - Setting the interrupt pace counts using the EMAC control module registers INTCONTROL, CORXIMAX, and COTXIMAX
  - Initializing the EMAC and MDIO modules
  - Enabling interrupts in the EMAC control module using the EMAC control module interrupt control registers CORXTHRESHEN, CORXEN, COTXEN, and COMISCEN.

The process of mapping the EMAC interrupts to the CPU is done through the Vectored Interrupt Manager (VIM). Once the interrupt is mapped to a CPU interrupt, general masking and unmasking of interrupts (to control reentrancy) should be done at the chip level by manipulating the interrupt core enable mask registers.

### **27.2.16.3 MDIO Module Initialization**

The MDIO module is used to initially configure and monitor one or more external PHY devices. Other than initializing the software state machine (details on this state machine can be found in the IEEE 802.3 standard), all that needs to be done for the MDIO module is to enable the MDIO engine and to configure the clock divider. To set the clock divider, supply an MDIO clock of 1 MHz. For example, if the peripheral clock is 50 MHz, the divider can be set to 50.

Both the state machine enable and the MDIO clock divider are controlled through the MDIO control register (CONTROL). If none of the potentially connected PHYs require the access preamble, the PREAMBLE bit in CONTROL can also be set to speed up PHY register access.

If the MDIO module is to operate on an interrupt basis, the interrupts can be enabled at this time using the MDIO user command complete interrupt mask set register (USERINTMASKSET) for register access and the MDIO user PHY select register (USERPHYSEL $n$ ) if a target PHY is already known.

Once the MDIO state machine has been initialized and enabled, it starts polling all 32 PHY addresses on the MDIO bus, looking for an active PHY. Since it can take up to 50  $\mu$ s to read one register, it can be some time before the MDIO module provides an accurate representation of whether a PHY is available. Also, a PHY can take up to 3 seconds to negotiate a link. Thus, it is advisable to run the MDIO software off a time-based event rather than polling.

For more information on PHY control registers, see your PHY device documentation.

#### 27.2.16.4 EMAC Module Initialization

The EMAC module is used to send and receive data packets over the network. This is done by maintaining up to eight transmit and receive descriptor queues. The EMAC module configuration must also be kept up-to-date based on PHY negotiation results returned from the MDIO module. Most of the work in developing an application or device driver for Ethernet is programming this module.

The following is the initialization procedure a device driver would follow to get the EMAC to the state where it is ready to receive and send Ethernet packets. Some of these steps are not necessary when performed immediately after device reset.

1. If enabled, clear the device interrupt enable bits in the EMAC control module interrupt control registers C0RXTHRESHEN, C0RXEN, C0TXEN, and C0MISCEN.
2. Clear the MAC control register (MACCONTROL), receive control register (RXCONTROL), and transmit control register (TXCONTROL) (not necessary immediately after reset).
3. Initialize all 16 header descriptor pointer registers (RX $n$ HDP and TX $n$ HDP) to 0.
4. Clear all 36 statistics registers by writing 0 (not necessary immediately after reset).
5. Setup the local Ethernet MAC address by programming the MAC index register (MACINDEX), MAC address high bytes register (MACADDRHI), and MAC address low bytes register (MACADDRLO). Be sure to program all eight MAC address registers - whether the receive channel is to be enabled or not. Duplicate the same MAC address across all unused channels. When using more than one receive channel, start with channel 0 and progress upwards.
6. If buffer flow control is to be enabled, initialize the receive channel  $n$  free buffer count registers (RX $n$ FREEBUFFER), receive channel  $n$  flow control threshold register (RX $n$ FLOWTHRESH), and receive filter low priority frame threshold register (RXFILTERLOWTHRESH).
7. Most device drivers open with no multicast addresses, so clear the MAC address hash registers (MACHASH1 and MACHASH2) to 0.
8. Write the receive buffer offset register (RXBUFFEROFFSET) value (typically zero).
9. Initially clear all unicast channels by writing FFh to the receive unicast clear register (RXUNICASTCLEAR). If unicast is desired, it can be enabled now by writing the receive unicast set register (RXUNICASTSET). Some drivers will default to unicast on device open while others will not.
10. Setup the receive multicast/broadcast/promiscuous channel enable register (RXMBPENABLE) with an initial configuration. The configuration is based on the current receive filter settings of the device driver. Some drivers may enable things like broadcast and multicast packets immediately, while others may not.
11. Set the appropriate configuration bits in MACCONTROL (do not set the GMIEN bit yet).
12. Clear all unused channel interrupt bits by writing the receive interrupt mask clear register (RXINTMASKCLEAR) and the transmit interrupt mask clear register (TXINTMASKCLEAR).
13. Enable the receive and transmit channel interrupt bits in the receive interrupt mask set register (RXINTMASKSET) and the transmit interrupt mask set register (TXINTMASKSET) for the channels to be used, and enable the HOSTMASK and STATMASK bits using the MAC interrupt mask set register (MACINTMASKSET).
14. Initialize the receive and transmit descriptor list queues.
15. Prepare receive by writing a pointer to the head of the receive buffer descriptor list to RX $n$ HDP.
16. Enable the receive and transmit DMA controllers by setting the RXEN bit in RXCONTROL and the TXEN bit in TXCONTROL. Then set the GMIEN bit in MACCONTROL.
17. Enable the device interrupt in EMAC control module registers C0RXTHRESHEN, C0RXEN, C0TXEN, and C0MISCEN.

## 27.2.17 Interrupt Support

### 27.2.17.1 EMAC Module Interrupt Events and Requests

The EMAC module generates 26 interrupt events:

- TXPEND $n$ : Transmit packet completion interrupt for transmit channels 0 through 7
- RXPEND $n$ : Receive packet completion interrupt for receive channels 0 through 7
- RXTRESHPEND $n$ : Receive packet completion interrupt for receive channels 0 through 7 when flow control is enabled and the number of free buffers is below the threshold
- STATPEND: Statistics interrupt
- HOSTPEND: Host error interrupt

#### 27.2.17.1.1 Transmit Packet Completion Interrupts

The transmit DMA engine has eight channels, with each channel having a corresponding interrupt (TXPEND $n$ ). The transmit interrupts are level interrupts that remain asserted until cleared by the CPU.

Each of the eight transmit channel interrupts may be individually enabled by setting the appropriate bit in the transmit interrupt mask set register (TXINTMASKSET) to 1. Each of the eight transmit channel interrupts may be individually disabled by clearing the appropriate bit by writing a 1 to the transmit interrupt mask clear register (TXINTMASKCLEAR). The raw and masked transmit interrupt status may be read by reading the transmit interrupt status (unmasked) register (TXINTSTATRAW) and the transmit interrupt status (masked) register (TXINTSTATMASKED), respectively.

When the EMAC completes the transmission of a packet, the EMAC issues an interrupt to the CPU (via the EMAC control module) when it writes the packet's last buffer descriptor address to the appropriate channel queue's transmit completion pointer located in the state RAM block. The interrupt is generated by the write when enabled by the interrupt mask, regardless of the value written.

Upon interrupt reception, the CPU processes one or more packets from the buffer chain and then acknowledges an interrupt by writing the address of the last buffer descriptor processed to the queue's associated transmit completion pointer in the transmit DMA state RAM.

The data written by the host (buffer descriptor address of the last processed buffer) is compared to the data in the register written by the EMAC port (address of last buffer descriptor used by the EMAC). If the two values are not equal (which means that the EMAC has transmitted more packets than the CPU has processed interrupts for), the transmit packet completion interrupt signal remains asserted. If the two values are equal (which means that the host has processed all packets that the EMAC has transferred), the pending interrupt is cleared. The value that the EMAC is expecting is found by reading the transmit channel  $n$  completion pointer register (TX $n$ CP).

The EMAC write to the completion pointer actually stores the value in the state RAM. The CPU written value does not actually change the register value. The host written value is compared to the register content (which was written by the EMAC) and if the two values are equal then the interrupt is removed; otherwise, the interrupt remains asserted. The host may process multiple packets prior to acknowledging an interrupt, or the host may acknowledge interrupts for every packet.

The application software must acknowledge the EMAC control module after processing packets by writing the appropriate CORX key to the EMAC End-Of-Interrupt Vector register (MACEOIVECTOR). See [Section 27.5.12](#) for the acknowledge key values.

#### 27.2.17.1.2 Receive Packet Completion Interrupts

The receive DMA engine has eight channels, which each channel having a corresponding interrupt (RXPEND $n$ ). The receive interrupts are level interrupts that remain asserted until cleared by the CPU.

Each of the eight receive channel interrupts may be individually enabled by setting the appropriate bit in the receive interrupt mask set register (RXINTMASKSET) to 1. Each of the eight receive channel interrupts may be individually disabled by clearing the appropriate bit by writing a 1 in the receive interrupt mask clear register (RXINTMASKCLEAR). The raw and masked receive interrupt status may be read by reading the receive interrupt status (unmasked) register (RXINTSTATRAW) and the receive interrupt status (masked) register (RXINTSTATMASKED), respectively.



When the EMAC completes a packet reception, the EMAC issues an interrupt to the CPU by writing the packet's last buffer descriptor address to the appropriate channel queue's receive completion pointer located in the state RAM block. The interrupt is generated by the write when enabled by the interrupt mask, regardless of the value written.

Upon interrupt reception, the CPU processes one or more packets from the buffer chain and then acknowledges one or more interrupt(s) by writing the address of the last buffer descriptor processed to the queue's associated receive completion pointer in the receive DMA state RAM.

The data written by the host (buffer descriptor address of the last processed buffer) is compared to the data in the register written by the EMAC (address of last buffer descriptor used by the EMAC). If the two values are not equal (which means that the EMAC has received more packets than the CPU has processed interrupts for), the receive packet completion interrupt signal remains asserted. If the two values are equal (which means that the host has processed all packets that the EMAC has received), the pending interrupt is de-asserted. The value that the EMAC is expecting is found by reading the receive channel  $n$  completion pointer register (RX $n$ CP).

The EMAC write to the completion pointer actually stores the value in the state RAM. The CPU written value does not actually change the register value. The host written value is compared to the register content (which was written by the EMAC) and if the two values are equal then the interrupt is removed; otherwise, the interrupt remains asserted. The host may process multiple packets prior to acknowledging an interrupt, or the host may acknowledge interrupts for every packet.

The application software must acknowledge the EMAC control module after processing packets by writing the appropriate C0TX key to the EMAC End-Of-Interrupt Vector register (MACEOIVECTOR). See [Section 27.5.12](#) for the acknowledge key values.

#### 27.2.17.1.3 Statistics Interrupt

The statistics level interrupt (STATPEND) is issued when any statistics value is greater than or equal to 8000 0000h, if enabled by setting the STATMASK bit in the MAC interrupt mask set register (MACINTMASKSET) to 1. The statistics interrupt is removed by writing to decrement any statistics value greater than 8000 0000h. As long as the most-significant bit of any statistics value is set, the interrupt remains asserted.

The application software must acknowledge the EMAC control module after receiving statistics interrupts by writing the appropriate COMISC key to the EMAC End-Of-Interrupt Vector register (MACEOIVECTOR). See [Section 27.5.12](#) for the acknowledge key values.

#### 27.2.17.1.4 Host Error Interrupt

The host error interrupt (HOSTPEND) is issued, if enabled, under error conditions dealing with the handling of buffer descriptors, detected during transmit or receive DMA transactions. The failure of the software application to supply properly formatted buffer descriptors results in this error. The error bit can only be cleared by resetting the EMAC module in hardware.

The host error interrupt is enabled by setting the HOSTMASK bit in the MAC interrupt mask set register (MACINTMASKSET) to 1. The host error interrupt is disabled by clearing the appropriate bit by writing a 1 in the MAC interrupt mask clear register (MACINTMASKCLEAR). The raw and masked host error interrupt status may be read by reading the MAC interrupt status (unmasked) register (MACINTSTATRAW) and the MAC interrupt status (masked) register (MACINTSTATMASKED), respectively.

The transmit host error conditions are:

- SOP error
- Ownership bit not set in SOP buffer
- Zero next buffer descriptor pointer with EOP
- Zero buffer pointer
- Zero buffer length
- Packet length error

The receive host error conditions are:

- Ownership bit not set in input buffer
- Zero buffer pointer

The application software must acknowledge the EMAC control module after receiving host error interrupts by writing the appropriate COMISC key to the EMAC End-Of-Interrupt Vector (MACEOIVECTOR). See [Section 27.5.12](#) for the acknowledge key values.

#### **27.2.17.1.5 Receive Threshold Interrupts**

Each of the eight receive channels have a corresponding receive threshold interrupt (RX $n$ THRESHPEND). The receive threshold interrupts are level interrupts that remain asserted until the triggering condition is cleared by the host. Each of the eight threshold interrupts may be individually enabled by setting to 1 the appropriate bit in the RXINTMASKSET register. Each of the eight channel interrupts may be individually disabled by clearing to zero the appropriate bit by writing a 1 in the receive interrupt mask clear register (RXINTMASKCLEAR). The raw and masked interrupt receive interrupt status may be read by reading the receive interrupt status (unmasked) register (RXINTSTATRAW) and the receive interrupt status (masked) register (RXINTSTATMASKED), respectively.

An RX $n$ THRESHPEND interrupt bit is asserted when enabled and when the channel's associated free buffer count (RX $n$ FREEBUFFER) is less than or equal to the channel's associated flow control threshold register (RX $n$ FLOWTHRESH). The receive threshold interrupts use the same free buffer count and threshold logic as does flow control, but the interrupts are independently enabled from flow control. The threshold interrupts are intended to give the host an indication that resources are running low for a particular channel(s).

The applications software must acknowledge the EMAC control module after receiving threshold interrupts by writing the appropriate CORXTHRESH key to the EMAC End-Of-Interrupt Vector (MACEOIVECTOR). See [Section 27.5.12](#) for the acknowledge key values.

#### **27.2.17.2 MDIO Module Interrupt Events and Requests**

The MDIO module generates two interrupt events:

- LINKINT0: Serial interface link change interrupt. Indicates a change in the state of the PHY link selected by the USERPHYSEL0 register
- USERINT0: Serial interface user command event complete interrupt selected by the USERACCESS0 register

##### **27.2.17.2.1 Link Change Interrupt**

The MDIO module asserts a link change interrupt (LINKINT0) if there is a change in the link state of the PHY corresponding to the address in the PHYADRMON bit in the MDIO register USERPHYSEL0, and if the LINKINTENB bit is also set in USERPHYSEL0. This interrupt event is also captured in the LINKINTRAW bit in the MDIO link status change interrupt register (LINKINTRAW). LINKINTRAW bits 0 and 1 correspond to USERPHYSEL0 and USERPHYSEL1, respectively.

When the interrupt is enabled and generated, the corresponding LINKINTMASKED bit is also set in the MDIO link status change interrupt register (LINKINTMASKED). The interrupt is cleared by writing back the same bit to LINKINTMASKED (write to clear).

The application software must acknowledge the EMAC control module after receiving MDIO interrupts by writing the appropriate COMISC key to the EMAC End-Of-Interrupt Vector (MACEOIVECTOR). See [Section 27.5.12](#) for the acknowledge key values.



### 27.2.17.2.2 User Access Completion Interrupt

When the GO bit in one of the MDIO register USERACCESS0 transitions from 1 to 0 (indicating completion of a user access) and the corresponding USERINTMASKSET bit in the MDIO user command complete interrupt mask set register (USERINTMASKSET) corresponding to USERACCESS0 is set, a user access completion interrupt (USERINT) is asserted. This interrupt event is also captured in the USERINTRAW bit in the MDIO user command complete interrupt register (USERINTRAW). USERINTRAW bits 0 and bit 1 correspond to USERACCESS0 and USERACCESS1, respectively.

When the interrupt is enabled and generated, the corresponding USERINTMASKED bit is also set in the MDIO user command complete interrupt register (USERINTMASKED). The interrupt is cleared by writing back the same bit to USERINTMASKED (write to clear).

The application software must acknowledge the EMAC control module after receiving MDIO interrupts by writing the appropriate COMISC key to the EMAC End-Of-Interrupt Vector (MACEOIVECTOR). See [Section 27.5.12](#) for the acknowledge key values.

### 27.2.17.3 Proper Interrupt Processing

All the interrupts signaled from the EMAC and MDIO modules are level driven, so if they remain active, their level remains constant; the CPU core may require edge- or pulse-triggered interrupts. In order to properly convert the level-driven interrupt signal to an edge- or pulse-triggered signal, the application software must make use of the interrupt control logic contained in the EMAC control module.

[Section 27.2.7.3](#) discusses the interrupt control contained in the EMAC control module. For safe interrupt processing, upon entry to the ISR, the software application should disable interrupts using the EMAC control module registers CORXTHRESHEN, CORXEN, COTXEN, COMISCEN, and then reenables them upon leaving the ISR. If any interrupt signals are active at that time, this creates another rising edge on the interrupt signal going to the CPU interrupt controller, thus triggering another interrupt. The EMAC control module also uses the EMAC control module registers INTCONTROL, COTXIMAX, and CORXIMAX to implement interrupt pacing. The application software must acknowledge the EMAC control module by writing the appropriate key to the EMAC End-Of-Interrupt Vector (MACEOIVECTOR). See [Section 27.5.12](#) for the acknowledge key values.

### 27.2.17.4 Interrupt Multiplexing

The EMAC control module combines different interrupt signals from both the EMAC and MDIO modules into four interrupt signals (CORXTHRESHPULSE, CORXPULSE, COTXPULSE, COMISCPULSE) that are routed to the Vectored Interrupt Manager (VIM). The VIM is capable of relaying all four interrupt signals to the CPU.

When an interrupt is generated, the reason for the interrupt can be read from the MAC input vector register (MACINVECTOR) located in the EMAC memory map. MACINVECTOR combines the status of the following 28 interrupt signals: TXPEND $n$ , RXPEND $n$ , RXTHRESHPEND $n$ , STATPEND, HOSTPEND, LINKINT0, and USERINT0.

For more details on the interrupt mapping, see your device-specific datasheet and VIM chapter of the Technical Reference Manual.

### 27.2.18 Power Management

Each of the three main components of the EMAC peripheral can be placed in a reduced-power mode to conserve power during periods of low activity. The peripheral clock to the EMAC peripheral is controlled by the processor Global Clock Module (GCM). The GCM allows the application to enable or disable the peripheral clock to the EMAC peripheral.

The power conservation modes available for each of the three components of the EMAC/MDIO peripheral are:

- *Idle/Disabled state.* This mode stops the clocks going to the peripheral, and prevents all the register accesses. After reenabling the peripheral from this idle state, all the registers values prior to setting into the disabled state are restored, and data transmission can proceed. No reinitialization is required.
- *System reset.* The EMAC peripheral is reset by the system reset signal output from the System module. Refer to the "Architecture" chapter of the Technical Reference Manual to identify the causes of a system reset. Upon a system reset, the registers are reset to their default value. When powering-up after a system reset, all the EMAC submodules need to be reinitialized before any data transmission can happen.

For more information on the use of the GCM, see your device-specific Technical Reference Manual.

### 27.2.19 Emulation Considerations

EMAC emulation control is implemented for compatibility with other peripherals. The SOFT and FREE bits in the emulation control register (EMCONTROL) allow EMAC operation to be suspended.

When the emulation suspend state is entered, the EMAC stops processing receive and transmit frames at the next frame boundary. Any frame currently in reception or transmission is completed normally without suspension. For transmission, any complete or partial frame in the transmit cell FIFO is transmitted. For receive, frames that are detected by the EMAC after the suspend state is entered are ignored. No statistics are kept for ignored frames.

Table 27-9 shows how the SOFT and FREE bits affect the operation of the emulation suspend.

---

**NOTE:** Emulation suspend has not been tested.

---

**Table 27-9. Emulation Control**

SOFT	FREE	Description
0	0	Normal operation
1	0	Emulation suspend
X	1	Normal operation

## 27.3 EMAC Control Module Registers

Table 27-10 lists the memory-mapped registers for the EMAC control module. The base address for these registers is 0xFCF78800.

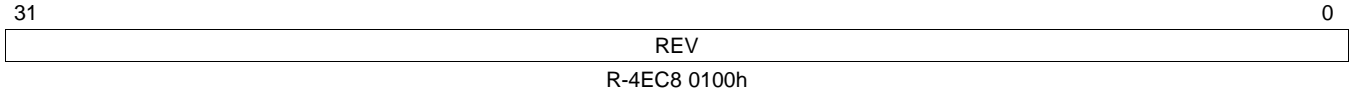
**Table 27-10. EMAC Control Module Registers**

Offset	Acronym	Register Description	Section
0h	REVID	EMAC Control Module Revision ID Register	<a href="#">Section 27.3.1</a>
4h	SOFTRESET	EMAC Control Module Software Reset Register	<a href="#">Section 27.3.2</a>
Ch	INTCONTROL	EMAC Control Module Interrupt Control Register	<a href="#">Section 27.3.3</a>
10h	C0RXTHRESHEN	EMAC Control Module Receive Threshold Interrupt Enable Register	<a href="#">Section 27.3.4</a>
14h	C0RXEN	EMAC Control Module Receive Interrupt Enable Register	<a href="#">Section 27.3.5</a>
18h	C0TXEN	EMAC Control Module Transmit Interrupt Enable Register	<a href="#">Section 27.3.6</a>
1Ch	C0MISCEN	EMAC Control Module Miscellaneous Interrupt Enable Register	<a href="#">Section 27.3.7</a>
20h - 3Ch	Reserved	Reserved	
40h	C0RXTHRESHSTAT	EMAC Control Module Receive Threshold Interrupt Status Register	<a href="#">Section 27.3.8</a>
44h	C0RXSTAT	EMAC Control Module Receive Interrupt Status Register	<a href="#">Section 27.3.9</a>
48h	C0TXSTAT	EMAC Control Module Transmit Interrupt Status Register	<a href="#">Section 27.3.10</a>
4Ch	C0MISCSTAT	EMAC Control Module Miscellaneous Interrupt Status Register	<a href="#">Section 27.3.11</a>
50h - 6Ch	Reserved	Reserved	
70h	C0RXIMAX	EMAC Control Module Receive Interrupts Per Millisecond Register	<a href="#">Section 27.3.12</a>
74h	C0TXIMAX	EMAC Control Module Transmit Interrupts Per Millisecond Register	<a href="#">Section 27.3.13</a>
78h - 84h	Reserved	Reserved	

**27.3.1 EMAC Control Module Revision ID Register (REVID)**

The EMAC control module revision ID register (REVID) is shown in [Figure 27-12](#) and described in [Table 27-11](#).

**Figure 27-12. EMAC Control Module Revision ID Register (REVID)**



LEGEND: R = Read only; -n = value after reset

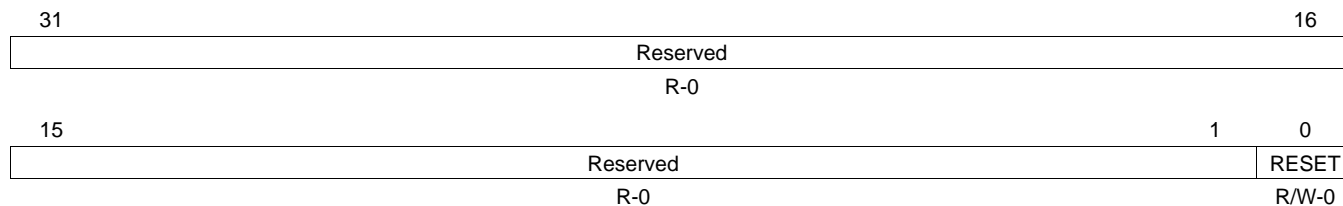
**Table 27-11. EMAC Control Module Revision ID Register (REVID) Field Descriptions**

Bit	Field	Value	Description
31-0	REV	4EC8 0100h	Identifies the EMAC Control Module revision. Current revision of the EMAC Control Module.

### 27.3.2 EMAC Control Module Software Reset Register (SOFTRESET)

The EMAC Control Module Software Reset Register (SOFTRESET) is shown in [Figure 27-13](#) and described in [Table 27-12](#).

**Figure 27-13. EMAC Control Module Software Reset Register (SOFTRESET)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

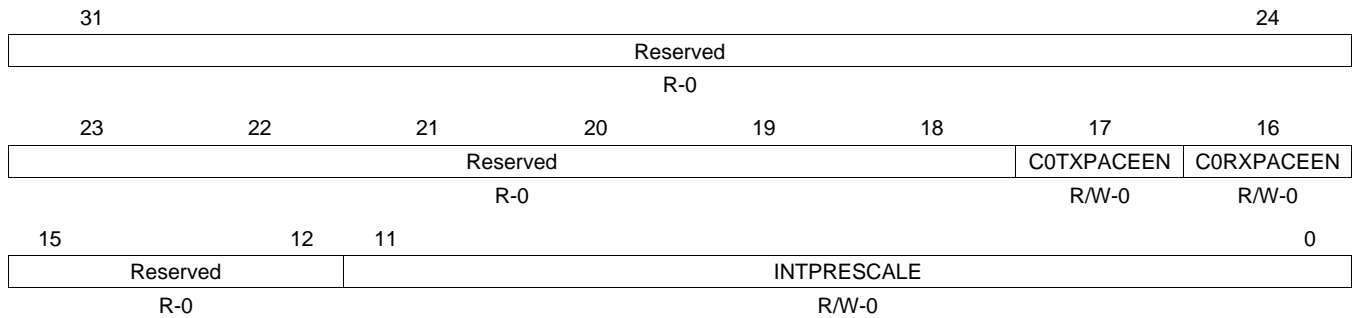
**Table 27-12. EMAC Control Module Software Reset Register (SOFTRESET)**

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	RESET		Software reset bit for the EMAC Control Module. Clears the interrupt status, control registers, and CPPI Ram on the clock cycle following a write of 1.
		0	No software reset.
		1	Perform a software reset.

### 27.3.3 EMAC Control Module Interrupt Control Register (INTCONTROL)

The EMAC control module interrupt control register (INTCONTROL) is shown in [Figure 27-14](#) and described in [Table 27-13](#). The settings in the INTCONTROL register are used in conjunction with the CnRXIMAX and CnTXIMAX registers.

**Figure 27-14. EMAC Control Module Interrupt Control Register (INTCONTROL)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

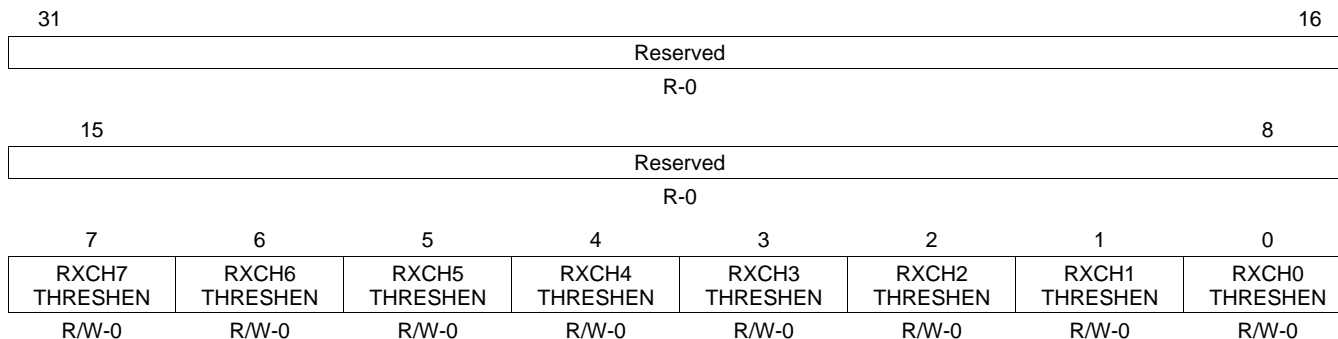
**Table 27-13. EMAC Control Module Interrupt Control Register (INTCONTROL)**

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17	COTXPACEEN	0 1	Enable pacing for TX interrupt pulse generation Pacing for TX interrupts disabled. Pacing for TX interrupts enabled.
16	CORXPACEEN	0 1	Enable pacing for RX interrupt pulse generation Pacing for RX interrupts disabled. Pacing for RX interrupts enabled.
15-12	Reserved	0	Reserved
11-0	INTPRESCALE	0-7FFh	Number of internal EMAC module reference clock periods within a 4 $\mu$ s time window (see your device-specific data manual for information).

### 27.3.4 EMAC Control Module Receive Threshold Interrupt Enable Registers (C0RXTHRESHEN)

The EMAC control module receive threshold interrupt enable register (C0RXTHRESHEN) is shown in Figure 27-15 and described in Table 27-14.

**Figure 27-15. EMAC Control Module Receive Threshold Interrupt Enable Register (C0RXTHRESHEN)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

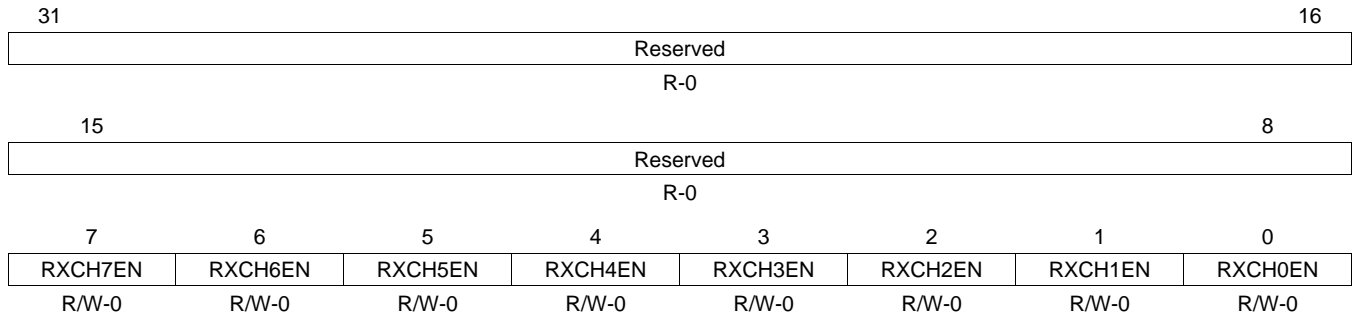
**Table 27-14. EMAC Control Module Receive Threshold Interrupt Enable Register (C0RXTHRESHEN)**

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	RXCH7THRESHEN	0 1	Enable C0RXTHRESHPULSE interrupt generation for RX Channel 7 C0RXTHRESHPULSE generation is disabled for RX Channel 7. C0RXTHRESHPULSE generation is enabled for RX Channel 7.
6	RXCH6THRESHEN	0 1	Enable C0RXTHRESHPULSE interrupt generation for RX Channel 6 C0RXTHRESHPULSE generation is disabled for RX Channel 6. C0RXTHRESHPULSE generation is enabled for RX Channel 6.
5	RXCH5THRESHEN	0 1	Enable C0RXTHRESHPULSE interrupt generation for RX Channel 5 C0RXTHRESHPULSE generation is disabled for RX Channel 5. C0RXTHRESHPULSE generation is enabled for RX Channel 5.
4	RXCH4THRESHEN	0 1	Enable C0RXTHRESHPULSE interrupt generation for RX Channel 4 C0RXTHRESHPULSE generation is disabled for RX Channel 4. C0RXTHRESHPULSE generation is enabled for RX Channel 4.
3	RXCH3THRESHEN	0 1	Enable C0RXTHRESHPULSE interrupt generation for RX Channel 3 C0RXTHRESHPULSE generation is disabled for RX Channel 3. C0RXTHRESHPULSE generation is enabled for RX Channel 3.
2	RXCH2THRESHEN	0 1	Enable C0RXTHRESHPULSE interrupt generation for RX Channel 2 C0RXTHRESHPULSE generation is disabled for RX Channel 2. C0RXTHRESHPULSE generation is enabled for RX Channel 2.
1	RXCH1THRESHEN	0 1	Enable C0RXTHRESHPULSE interrupt generation for RX Channel 1 C0RXTHRESHPULSE generation is disabled for RX Channel 1. C0RXTHRESHPULSE generation is enabled for RX Channel 1.
0	RXCH0THRESHEN	0 1	Enable C0RXTHRESHPULSE interrupt generation for RX Channel 0 C0RXTHRESHPULSE generation is disabled for RX Channel 0. C0RXTHRESHPULSE generation is enabled for RX Channel 0.

### 27.3.5 EMAC Control Module Receive Interrupt Enable Registers (C0RXEN)

The EMAC control module receive interrupt enable register (C0RXEN) is shown in [Figure 27-16](#) and described in [Table 27-15](#)

**Figure 27-16. EMAC Control Module Receive Interrupt Enable Register (C0RXEN)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 27-15. EMAC Control Module Receive Interrupt Enable Register (C0RXEN)**

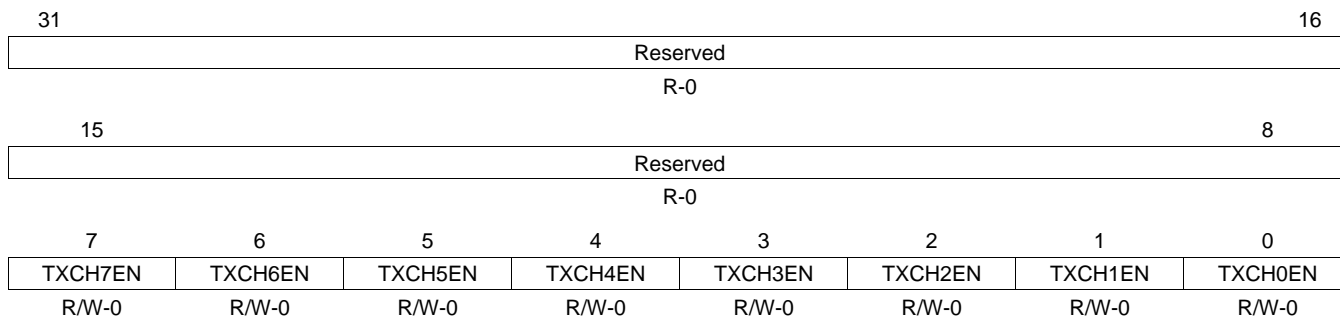
Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	RXCH7EN	0 1	Enable C0RXPULSE interrupt generation for RX Channel 7 C0RXPULSE generation is disabled for RX Channel 7. C0RXPULSE generation is enabled for RX Channel 7.
6	RXCH6EN	0 1	Enable C0RXPULSE interrupt generation for RX Channel 6 C0RXPULSE generation is disabled for RX Channel 6. C0RXPULSE generation is enabled for RX Channel 6.
5	RXCH5EN	0 1	Enable C0RXPULSE interrupt generation for RX Channel 5 C0RXPULSE generation is disabled for RX Channel 5. C0RXPULSE generation is enabled for RX Channel 5.
4	RXCH4EN	0 1	Enable C0RXPULSE interrupt generation for RX Channel 4 C0RXPULSE generation is disabled for RX Channel 4. C0RXPULSE generation is enabled for RX Channel 4.
3	RXCH3EN	0 1	Enable C0RXPULSE interrupt generation for RX Channel 3 C0RXPULSE generation is disabled for RX Channel 3. C0RXPULSE generation is enabled for RX Channel 3.
2	RXCH2EN	0 1	Enable C0RXPULSE interrupt generation for RX Channel 2 C0RXPULSE generation is disabled for RX Channel 2. C0RXPULSE generation is enabled for RX Channel 2.
1	RXCH1EN	0 1	Enable C0RXPULSE interrupt generation for RX Channel 1 C0RXPULSE generation is disabled for RX Channel 1. C0RXPULSE generation is enabled for RX Channel 1.
0	RXCH0EN	0 1	Enable C0RXPULSE interrupt generation for RX Channel 0 C0RXPULSE generation is disabled for RX Channel 0. C0RXPULSE generation is enabled for RX Channel 0.



### 27.3.6 EMAC Control Module Transmit Interrupt Enable Registers (C0TXEN)

The EMAC control module transmit interrupt enable register (C0TXEN) is shown in [Figure 27-17](#) and described in [Table 27-16](#)

**Figure 27-17. EMAC Control Module Transmit Interrupt Enable Register (C0TXEN)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

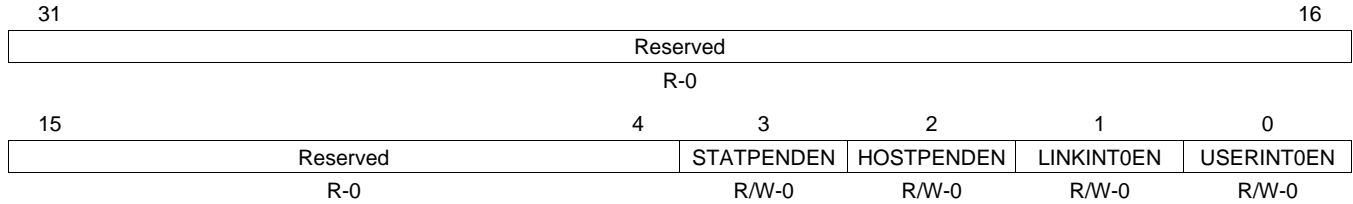
**Table 27-16. EMAC Control Module Transmit Interrupt Enable Register (C0TXEN)**

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	TXCH7EN	0 1	Enable C0TXPULSE interrupt generation for TX Channel 7 C0TXPULSE generation is disabled for TX Channel 7. C0TXPULSE generation is enabled for TX Channel 7.
6	TXCH6EN	0 1	Enable C0TXPULSE interrupt generation for TX Channel 6 C0TXPULSE generation is disabled for TX Channel 6. C0TXPULSE generation is enabled for TX Channel 6.
5	TXCH5EN	0 1	Enable C0TXPULSE interrupt generation for TX Channel 5 C0TXPULSE generation is disabled for TX Channel 5. C0TXPULSE generation is enabled for TX Channel 5.
4	TXCH4EN	0 1	Enable C0TXPULSE interrupt generation for TX Channel 4 C0TXPULSE generation is disabled for TX Channel 4. C0TXPULSE generation is enabled for TX Channel 4.
3	TXCH3EN	0 1	Enable C0TXPULSE interrupt generation for TX Channel 3 C0TXPULSE generation is disabled for TX Channel 3. C0TXPULSE generation is enabled for TX Channel 3.
2	TXCH2EN	0 1	Enable C0TXPULSE interrupt generation for TX Channel 2 C0TXPULSE generation is disabled for TX Channel 2. C0TXPULSE generation is enabled for TX Channel 2.
1	TXCH1EN	0 1	Enable C0TXPULSE interrupt generation for TX Channel 1 C0TXPULSE generation is disabled for TX Channel 1. C0TXPULSE generation is enabled for TX Channel 1.
0	TXCH0EN	0 1	Enable C0TXPULSE interrupt generation for TX Channel 0 C0TXPULSE generation is disabled for TX Channel 0. C0TXPULSE generation is enabled for TX Channel 0.

### 27.3.7 EMAC Control Module Miscellaneous Interrupt Enable Registers (COMISCEN)

The EMAC control module miscellaneous interrupt enable register (COMISCEN) is shown in [Figure 27-18](#) and described in [Table 27-17](#)

**Figure 27-18. EMAC Control Module Miscellaneous Interrupt Enable Register (COMISCEN)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

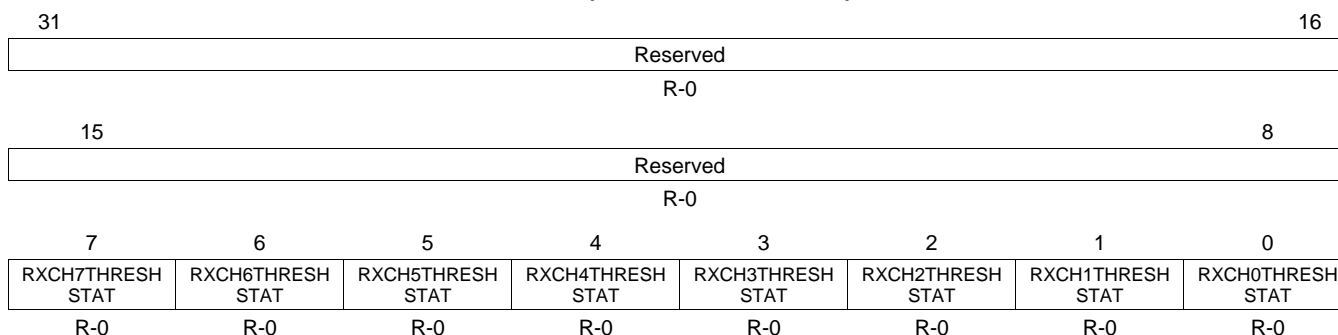
**Table 27-17. EMAC Control Module Miscellaneous Interrupt Enable Register (COMISCEN)**

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3	STATPENDEN	0	Enable COMISCPULSE interrupt generation when EMAC statistics interrupts are generated
		1	COMISCPULSE generation is disabled for EMAC STATPEND interrupts.
2	HOSTPENDEN	0	Enable COMISCPULSE interrupt generation when EMAC host interrupts are generated
		1	COMISCPULSE generation is disabled for EMAC HOSTPEND interrupts.
1	LINKINT0EN	0	Enable COMISCPULSE interrupt generation when MDIO LINKINT0 interrupts (corresponding to USERPHYSEL0) are generated
		1	COMISCPULSE generation is disabled for MDIO LINKINT0 interrupts.
0	USERINT0EN	0	Enable COMISCPULSE interrupt generation when MDIO USERINT0 interrupts (corresponding to USERACCESS0) are generated
		1	COMISCPULSE generation is disabled for MDIO USERINT0.

### 27.3.8 EMAC Control Module Receive Threshold Interrupt Status Registers (C0RXTHRESHSTAT)

The EMAC control module receive threshold interrupt status register (C0RXTHRESHSTAT) is shown in Figure 27-19 and described in Table 27-18

**Figure 27-19. EMAC Control Module Receive Threshold Interrupt Status Register  
(C0RXTHRESHSTAT)**



LEGEND: R = Read only; -n = value after reset

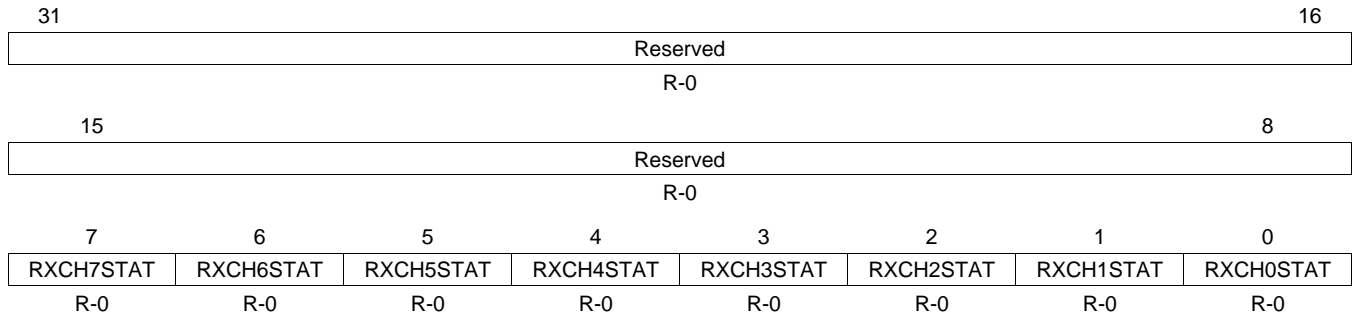
**Table 27-18. EMAC Control Module Receive Threshold Interrupt Status Register  
(C0RXTHRESHSTAT)**

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	RXCH7THRESHSTAT	0 1	Interrupt status for RX Channel 7 masked by the C0RXTHRESHEN register RX Channel 7 does not satisfy conditions to generate a C0RXTHRESHPULSE interrupt. RX Channel 7 satisfies conditions to generate a C0RXTHRESHPULSE interrupt.
6	RXCH6THRESHSTAT	0 1	Interrupt status for RX Channel 6 masked by the C0RXTHRESHEN register RX Channel 6 does not satisfy conditions to generate a C0RXTHRESHPULSE interrupt. RX Channel 6 satisfies conditions to generate a C0RXTHRESHPULSE interrupt.
5	RXCH5THRESHSTAT	0 1	Interrupt status for RX Channel 5 masked by the C0RXTHRESHEN register RX Channel 5 does not satisfy conditions to generate a C0RXTHRESHPULSE interrupt. RX Channel 5 satisfies conditions to generate a C0RXTHRESHPULSE interrupt.
4	RXCH4THRESHSTAT	0 1	Interrupt status for RX Channel 4 masked by the C0RXTHRESHEN register RX Channel 4 does not satisfy conditions to generate a C0RXTHRESHPULSE interrupt. RX Channel 4 satisfies conditions to generate a C0RXTHRESHPULSE interrupt.
3	RXCH3THRESHSTAT	0 1	Interrupt status for RX Channel 3 masked by the C0RXTHRESHEN register RX Channel 3 does not satisfy conditions to generate a C0RXTHRESHPULSE interrupt. RX Channel 3 satisfies conditions to generate a C0RXTHRESHPULSE interrupt.
2	RXCH2THRESHSTAT	0 1	Interrupt status for RX Channel 2 masked by the C0RXTHRESHEN register RX Channel 2 does not satisfy conditions to generate a C0RXTHRESHPULSE interrupt. RX Channel 2 satisfies conditions to generate a C0RXTHRESHPULSE interrupt.
1	RXCH1THRESHSTAT	0 1	Interrupt status for RX Channel 1 masked by the C0RXTHRESHEN register RX Channel 1 does not satisfy conditions to generate a C0RXTHRESHPULSE interrupt. RX Channel 1 satisfies conditions to generate a C0RXTHRESHPULSE interrupt.
0	RXCH0THRESHSTAT	0 1	Interrupt status for RX Channel 0 masked by the C0RXTHRESHEN register RX Channel 0 does not satisfy conditions to generate a C0RXTHRESHPULSE interrupt. RX Channel 0 satisfies conditions to generate a C0RXTHRESHPULSE interrupt.

### 27.3.9 EMAC Control Module Receive Interrupt Status Registers (C0RXSTAT)

The EMAC control module receive interrupt status register (C0RXSTAT) is shown in [Figure 27-20](#) and described in [Table 27-19](#)

**Figure 27-20. EMAC Control Module Receive Interrupt Status Register (C0RXSTAT)**



LEGEND: R = Read only; -n = value after reset

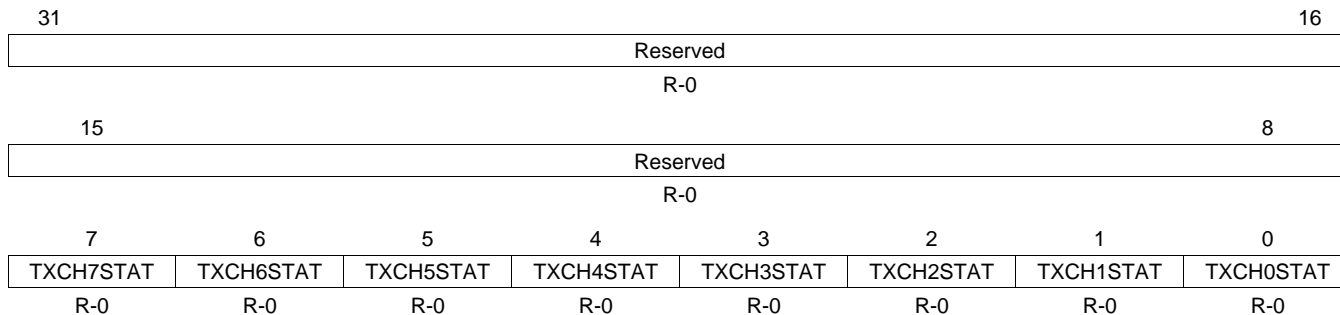
**Table 27-19. EMAC Control Module Receive Interrupt Status Register (C0RXSTAT)**

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	RXCH7STAT	0 1	Interrupt status for RX Channel 7 masked by the C0RXEN register RX Channel 7 does not satisfy conditions to generate a C0RXPULSE interrupt. RX Channel 7 satisfies conditions to generate a C0RXPULSE interrupt.
6	RXCH6STAT	0 1	Interrupt status for RX Channel 6 masked by the C0RXEN register RX Channel 6 does not satisfy conditions to generate a C0RXPULSE interrupt. RX Channel 6 satisfies conditions to generate a C0RXPULSE interrupt.
5	RXCH5STAT	0 1	Interrupt status for RX Channel 5 masked by the C0RXEN register RX Channel 5 does not satisfy conditions to generate a C0RXPULSE interrupt. RX Channel 5 satisfies conditions to generate a C0RXPULSE interrupt.
4	RXCH4STAT	0 1	Interrupt status for RX Channel 4 masked by the C0RXEN register RX Channel 4 does not satisfy conditions to generate a C0RXPULSE interrupt. RX Channel 4 satisfies conditions to generate a C0RXPULSE interrupt.
3	RXCH3STAT	0 1	Interrupt status for RX Channel 3 masked by the C0RXEN register RX Channel 3 does not satisfy conditions to generate a C0RXPULSE interrupt. RX Channel 3 satisfies conditions to generate a C0RXPULSE interrupt.
2	RXCH2STAT	0 1	Interrupt status for RX Channel 2 masked by the C0RXEN register RX Channel 2 does not satisfy conditions to generate a C0RXPULSE interrupt. RX Channel 2 satisfies conditions to generate a C0RXPULSE interrupt.
1	RXCH1STAT	0 1	Interrupt status for RX Channel 1 masked by the C0RXEN register RX Channel 1 does not satisfy conditions to generate a C0RXPULSE interrupt. RX Channel 1 satisfies conditions to generate a C0RXPULSE interrupt.
0	RXCH0STAT	0 1	Interrupt status for RX Channel 0 masked by the C0RXEN register RX Channel 0 does not satisfy conditions to generate a C0RXPULSE interrupt. RX Channel 0 satisfies conditions to generate a C0RXPULSE interrupt.

### 27.3.10 EMAC Control Module Transmit Interrupt Status Registers (C0TXSTAT)

The EMAC control module transmit interrupt status register (C0TXSTAT) is shown in [Figure 27-21](#) and described in [Table 27-20](#)

**Figure 27-21. EMAC Control Module Transmit Interrupt Status Register (C0TXSTAT)**



LEGEND: R = Read only; -n = value after reset

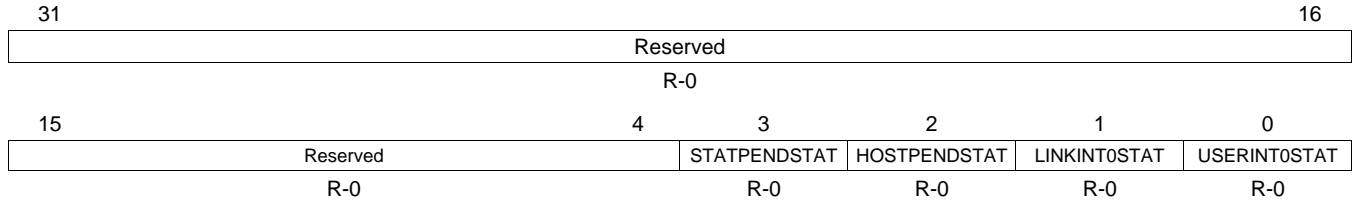
**Table 27-20. EMAC Control Module Transmit Interrupt Status Register (C0TXSTAT)**

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	TXCH7STAT	0 1	Interrupt status for TX Channel 7 masked by the C0TXEN register TX Channel 7 does not satisfy conditions to generate a C0TXPULSE interrupt. TX Channel 7 satisfies conditions to generate a C0TXPULSE interrupt.
6	TXCH6STAT	0 1	Interrupt status for TX Channel 6 masked by the C0TXEN register TX Channel 6 does not satisfy conditions to generate a C0TXPULSE interrupt. TX Channel 6 satisfies conditions to generate a C0TXPULSE interrupt.
5	TXCH5STAT	0 1	Interrupt status for TX Channel 5 masked by the C0TXEN register TX Channel 5 does not satisfy conditions to generate a C0TXPULSE interrupt. TX Channel 5 satisfies conditions to generate a C0TXPULSE interrupt.
4	TXCH4STAT	0 1	Interrupt status for TX Channel 4 masked by the C0TXEN register TX Channel 4 does not satisfy conditions to generate a C0TXPULSE interrupt. TX Channel 4 satisfies conditions to generate a C0TXPULSE interrupt.
3	TXCH3STAT	0 1	Interrupt status for TX Channel 3 masked by the C0TXEN register TX Channel 3 does not satisfy conditions to generate a C0TXPULSE interrupt. TX Channel 3 satisfies conditions to generate a C0TXPULSE interrupt.
2	TXCH2STAT	0 1	Interrupt status for TX Channel 2 masked by the C0TXEN register TX Channel 2 does not satisfy conditions to generate a C0TXPULSE interrupt. TX Channel 2 satisfies conditions to generate a C0TXPULSE interrupt.
1	TXCH1STAT	0 1	Interrupt status for TX Channel 1 masked by the C0TXEN register TX Channel 1 does not satisfy conditions to generate a C0TXPULSE interrupt. TX Channel 1 satisfies conditions to generate a C0TXPULSE interrupt.
0	TXCH0STAT	0 1	Interrupt status for TX Channel 0 masked by the C0TXEN register TX Channel 0 does not satisfy conditions to generate a C0TXPULSE interrupt. TX Channel 0 satisfies conditions to generate a C0TXPULSE interrupt.

### 27.3.11 EMAC Control Module Miscellaneous Interrupt Status Registers (C0MISCSTAT)

The EMAC control module miscellaneous interrupt status register (C0MISCSTAT) is shown in [Figure 27-22](#) and described in [Table 27-21](#)

**Figure 27-22. EMAC Control Module Miscellaneous Interrupt Status Register (C0MISCSTAT)**



LEGEND: R = Read only; -n = value after reset

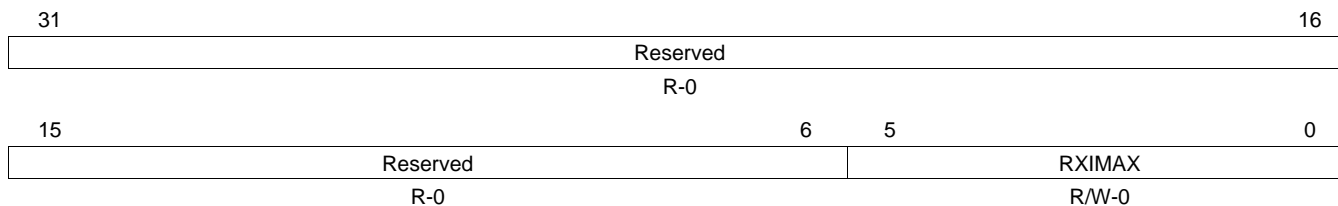
**Table 27-21. EMAC Control Module Miscellaneous Interrupt Status Register (C0MISCSTAT)**

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3	STATPENDSTAT	0	Interrupt status for EMAC STATPEND masked by the C0MISCEN register EMAC STATPEND does not satisfy conditions to generate a C0MISCPULSE interrupt.
		1	EMAC STATPEND satisfies conditions to generate a C0MISCPULSE interrupt.
2	HOSTPENDSTAT	0	Interrupt status for EMAC HOSTPEND masked by the C0MISCEN register EMAC HOSTPEND does not satisfy conditions to generate a C0MISCPULSE interrupt.
		1	EMAC HOSTPEND satisfies conditions to generate a C0MISCPULSE interrupt.
1	LINKINT0STAT	0	Interrupt status for MDIO LINKINT0 masked by the C0MISCEN register MDIO LINKINT0 does not satisfy conditions to generate a C0MISCPULSE interrupt.
		1	MDIO LINKINT0 satisfies conditions to generate a C0MISCPULSE interrupt.
0	USERINT0STAT	0	Interrupt status for MDIO USERINT0 masked by the C0MISCEN register MDIO USERINT0 does not satisfy conditions to generate a C0MISCPULSE interrupt.
		1	MDIO USERINT0 satisfies conditions to generate a C0MISCPULSE interrupt.

### 27.3.12 EMAC Control Module Receive Interrupts Per Millisecond Registers (C0RXIMAX)

The EMAC control module receive interrupts per millisecond register (C0RXIMAX) is shown in [Figure 27-23](#) and described in [Table 27-22](#)

**Figure 27-23. EMAC Control Module Receive Interrupts Per Millisecond Register (C0RXIMAX)**



LEGEND: R = Read only; R/W = Read/Write; -n = value after reset

**Table 27-22. EMAC Control Module Receive Interrupts Per Millisecond Register (C0RXIMAX)**

Bit	Field	Value	Description
31-6	Reserved	0	Reserved
5-0	RXIMAX	2-3Fh	RXIMAX is the desired number of C0RXPULSE interrupts generated per millisecond when C0RXPACEEN is enabled in INTCONTROL.

The pacing mechanism can be described by the following pseudo-code:

```

while(1) {
    interrupt_count = 0;

    /* Count interrupts over a lms window */
    for(i = 0; i < INTCONTROL[INTPRESCALE]*250; i++) {
        interrupt_count += NEW_INTERRUPT_EVENTS();

        if(i < INTCONTROL[INTPRESCALE]*pace_counter)
            BLOCK_EMAC_INTERRUPTS();
        else
            ALLOW_EMAC_INTERRUPTS();
    }

    ALLOW_EMAC_INTERRUPTS();

    if(interrupt_count > 2*RXIMAX)
        pace_counter = 255;
    else if(interrupt_count > 1.5*RXIMAX)
        pace_counter = previous_pace_counter*2 + 1;
    else if(interrupt_count > 1.0*RXIMAX)
        pace_counter = previous_pace_counter + 1;
    else if(interrupt_count > 0.5*RXIMAX)
        pace_counter = previous_pace_counter - 1;
    else if(interrupt_count != 0)
        pace_counter = previous_pace_counter/2;
    else
        pace_counter = 0;

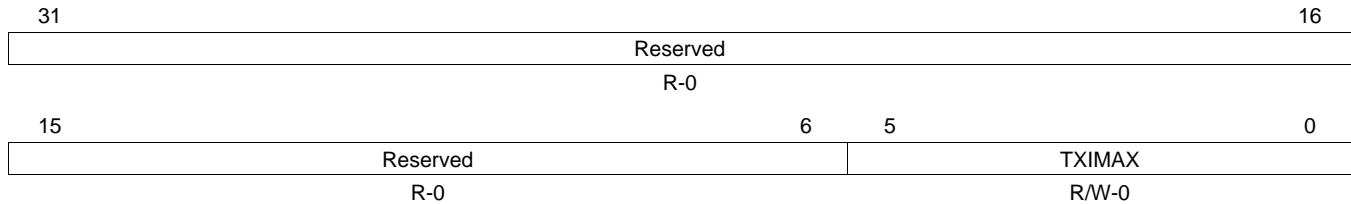
    previous_pace_counter = pace_counter;
}

```

### 27.3.13 EMAC Control Module Transmit Interrupts Per Millisecond Registers (C0TXIMAX)

The EMAC control module transmit interrupts per millisecond register (C0TXIMAX) is shown in [Figure 27-24](#) and described in [Table 27-23](#)

**Figure 27-24. EMAC Control Module Transmit Interrupts Per Millisecond Register (C0TXIMAX)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 27-23. EMAC Control Module Transmit Interrupts Per Millisecond Register (C0TXIMAX)**

Bit	Field	Value	Description
31-6	Reserved	0	Reserved
5-0	TXIMAX	2-3Fh	TXIMAX is the desired number of C0TXPULSE interrupts generated per millisecond when C0TXPACEEN is enabled in INTCONTROL.

The pacing mechanism can be described by the following pseudo-code:

```

while(1) {
    interrupt_count = 0;

    /* Count interrupts over a lms window */
    for(i = 0; i < INTCONTROL[INTPRESCALE]*250; i++) {
        interrupt_count += NEW_INTERRUPT_EVENTS();

        if(i < INTCONTROL[INTPRESCALE]*pace_counter)
            BLOCK_EMAC_INTERRUPTS();
        else
            ALLOW_EMAC_INTERRUPTS();
    }

    ALLOW_EMAC_INTERRUPTS();

    if(interrupt_count > 2*TXIMAX)
        pace_counter = 255;
    else if(interrupt_count > 1.5*TXIMAX)
        pace_counter = previous_pace_counter*2 + 1;
    else if(interrupt_count > 1.0*TXIMAX)
        pace_counter = previous_pace_counter + 1;
    else if(interrupt_count > 0.5*TXIMAX)
        pace_counter = previous_pace_counter - 1;
    else if(interrupt_count != 0)
        pace_counter = previous_pace_counter/2;
    else
        pace_counter = 0;

    previous_pace_counter = pace_counter;
}

```



## 27.4 MDIO Registers

Table 27-24 lists the memory-mapped registers for the MDIO module. The base address for these registers is FCF7 8900h.

**Table 27-24. Management Data Input/Output (MDIO) Registers**

Offset	Acronym	Register Description	Section
0h	REVID	MDIO Revision ID Register	<a href="#">Section 27.4.1</a>
4h	CONTROL	MDIO Control Register	<a href="#">Section 27.4.2</a>
8h	ALIVE	PHY Alive Status register	<a href="#">Section 27.4.3</a>
Ch	LINK	PHY Link Status Register	<a href="#">Section 27.4.4</a>
10h	LINKINTRAW	MDIO Link Status Change Interrupt (Unmasked) Register	<a href="#">Section 27.4.5</a>
14h	LINKINTMASKED	MDIO Link Status Change Interrupt (Masked) Register	<a href="#">Section 27.4.6</a>
20h	USERINTRAW	MDIO User Command Complete Interrupt (Unmasked) Register	<a href="#">Section 27.4.7</a>
24h	USERINTMASKED	MDIO User Command Complete Interrupt (Masked) Register	<a href="#">Section 27.4.8</a>
28h	USERINTMASKSET	MDIO User Command Complete Interrupt Mask Set Register	<a href="#">Section 27.4.9</a>
2Ch	USERINTMASKCLEAR	MDIO User Command Complete Interrupt Mask Clear Register	<a href="#">Section 27.4.10</a>
80h	USERACCESS0	MDIO User Access Register 0	<a href="#">Section 27.4.11</a>
84h	USERPHYSEL0	MDIO User PHY Select Register 0	<a href="#">Section 27.4.12</a>
88h	USERACCESS1	MDIO User Access Register 1	<a href="#">Section 27.4.13</a>
8Ch	USERPHYSEL1	MDIO User PHY Select Register 1	<a href="#">Section 27.4.14</a>

### 27.4.1 MDIO Revision ID Register (REVID)

The MDIO revision ID register (REVID) is shown in [Figure 27-25](#) and described in [Table 27-25](#).

**Figure 27-25. MDIO Revision ID Register (REVID)**



LEGEND: R = Read only; -n = value after reset

**Table 27-25. MDIO Revision ID Register (REVID) Field Descriptions**

Bit	Field	Value	Description
31-0	REV	0007 0104h	Identifies the MDIO Module revision. Current revision of the MDIO Module.

## 27.4.2 MDIO Control Register (CONTROL)

The MDIO control register (CONTROL) is shown in [Figure 27-26](#) and described in [Table 27-26](#).

**Figure 27-26. MDIO Control Register (CONTROL)**

31	30	29	28	24	23	21	20	19	18	17	16
IDLE	ENABLE	Rsvd	HIGHEST_USER_CHANNEL	Reserved		PREAMBLE	FAULT	FAULTENB	Reserved		
R-1	R/W-0	R-0	R-1	R-0		R/W-0	R/W1C-0	R/W-0	R-0		
15											0
CLKDIV											
R/W-FFh											

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear (writing a 0 has no effect); -n = value after reset

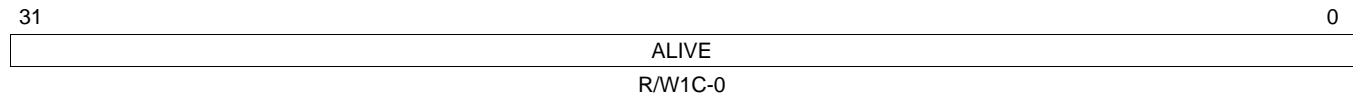
**Table 27-26. MDIO Control Register (CONTROL) Field Descriptions**

Bit	Field	Value	Description
31	IDLE	0 1	State machine IDLE status bit. State machine is not in idle state. State machine is in idle state.
30	ENABLE	0 1	State machine enable control bit. If the MDIO state machine is active at the time it is disabled, it will complete the current operation before halting and setting the idle bit. Disables the MDIO state machine. Enable the MDIO state machine.
29	Reserved	0	Reserved
28-24	HIGHEST_USER_CHANNEL	0-1Fh	Highest user channel that is available in the module. It is currently set to 1. This implies that MDIOUserAccess1 is the highest available user access channel.
23-21	Reserved	0	Reserved
20	PREAMBLE	0 1	Preamble disable Standard MDIO preamble is used. Disables this device from sending MDIO frame preambles.
19	FAULT	0 1	Fault indicator. This bit is set to 1 if the MDIO pins fail to read back what the device is driving onto them. This indicates a physical layer fault and the module state machine is reset. Writing a 1 to this bit clears this bit, writing a 0 has no effect. No failure Physical layer fault; the MDIO state machine is reset.
18	FAULTENB	0 1	Fault detect enable. This bit has to be set to 1 to enable the physical layer fault detection. Disables the physical layer fault detection. Enables the physical layer fault detection.
17-16	Reserved	0	Reserved
15-0	CLKDIV	0-FFFFh	Clock Divider bits. This field specifies the division ratio between the peripheral clock and the frequency of MDIO_CLK. MDIO_CLK is disabled when CLKDIV is cleared to 0. MDIO_CLK frequency = peripheral clock frequency/(CLKDIV + 1).

### 27.4.3 PHY Acknowledge Status Register (ALIVE)

The PHY acknowledge status register (ALIVE) is shown in [Figure 27-27](#) and described in [Table 27-27](#).

**Figure 27-27. PHY Acknowledge Status Register (ALIVE)**



LEGEND: R/W = Read/Write; W1C = Write 1 to clear (writing a 0 has no effect); -n = value after reset

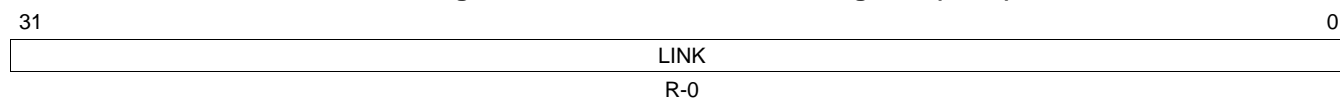
**Table 27-27. PHY Acknowledge Status Register (ALIVE) Field Descriptions**

Bit	Field	Value	Description
31-0	ALIVE		MDIO Alive bits. Each of the 32 bits of this register is set if the most recent access to the PHY with address corresponding to the register bit number was acknowledged by the PHY; the bit is reset if the PHY fails to acknowledge the access. Both the user and polling accesses to a PHY will cause the corresponding alive bit to be updated. The alive bits are only meant to be used to give an indication of the presence or not of a PHY with the corresponding address. Writing a 1 to any bit will clear it, writing a 0 has no effect.
		0	The PHY fails to acknowledge the access.
		1	The most recent access to the PHY with an address corresponding to the register bit number was acknowledged by the PHY.

### 27.4.4 PHY Link Status Register (LINK)

The PHY link status register (LINK) is shown in [Figure 27-28](#) and described in [Table 27-28](#).

**Figure 27-28. PHY Link Status Register (LINK)**



LEGEND: R = Read only; -n = value after reset

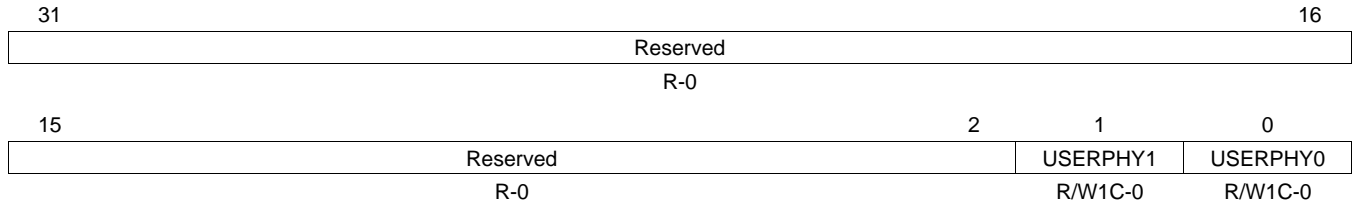
**Table 27-28. PHY Link Status Register (LINK) Field Descriptions**

Bit	Field	Value	Description
31-0	LINK		MDIO Link state bits. This register is updated after a read of the generic status register of a PHY. The bit is set if the PHY with the corresponding address has link and the PHY acknowledges the read transaction. The bit is reset if the PHY indicates it does not have link or fails to acknowledge the read transaction. Writes to the register have no effect.
		0	The PHY indicates it does not have a link or fails to acknowledge the read transaction
		1	The PHY with the corresponding address has a link and the PHY acknowledges the read transaction.

**27.4.5 MDIO Link Status Change Interrupt (Unmasked) Register (LINKINTRAW)**

The MDIO link status change interrupt (unmasked) register (LINKINTRAW) is shown in Figure 27-29 and described in Table 27-29.

**Figure 27-29. MDIO Link Status Change Interrupt (Unmasked) Register (LINKINTRAW)**



LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear (writing a 0 has no effect); -n = value after reset

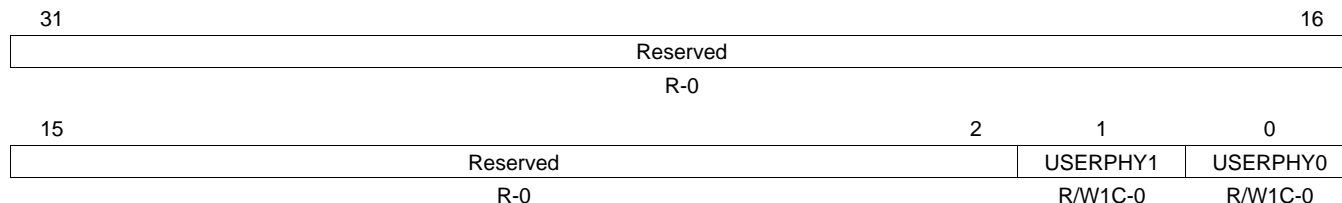
**Table 27-29. MDIO Link Status Change Interrupt (Unmasked) Register (LINKINTRAW) Field Descriptions**

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	USERPHY1	0	No MDIO link change event.
		1	An MDIO link change event (change in the LINK register) corresponding to the PHY address in MDIO user PHY select register USERPHYSEL1
0	USERPHY0	0	No MDIO link change event.
		1	An MDIO link change event (change in the LINK register) corresponding to the PHY address in MDIO user PHY select register USERPHYSEL0

### 27.4.6 MDIO Link Status Change Interrupt (Masked) Register (LINKINTMASKED)

The MDIO link status change interrupt (masked) register (LINKINTMASKED) is shown in [Figure 27-30](#) and described in [Table 27-30](#).

**Figure 27-30. MDIO Link Status Change Interrupt (Masked) Register (LINKINTMASKED)**



LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear (writing a 0 has no effect); -n = value after reset

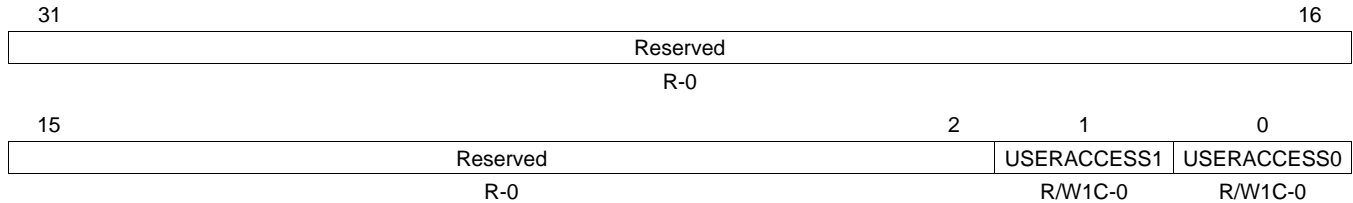
**Table 27-30. MDIO Link Status Change Interrupt (Masked) Register (LINKINTMASKED)  
Field Descriptions**

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	USERPHY1	0	MDIO Link change interrupt, masked value. When asserted, the bit indicates that there was an MDIO link change event (that is, change in the LINK register) corresponding to the PHY address in USERPHYSEL1 and the corresponding LINKINTENB bit was set. Writing a 1 will clear the event, writing a 0 has no effect.
		1	No MDIO link change event.
		1	An MDIO link change event (change in the LINK register) corresponding to the PHY address in MDIO user PHY select register USERPHYSEL1 and the LINKINTENB bit in USERPHYSEL1 is set to 1.
0	USERPHY0	0	MDIO Link change interrupt, masked value. When asserted, the bit indicates that there was an MDIO link change event (that is, change in the LINK register) corresponding to the PHY address in USERPHYSEL0 and the corresponding LINKINTENB bit was set. Writing a 1 will clear the event, writing a 0 has no effect.
		1	No MDIO link change event.
		1	An MDIO link change event (change in the LINK register) corresponding to the PHY address in MDIO user PHY select register USERPHYSEL0 and the LINKINTENB bit in USERPHYSEL0 is set to 1.

### 27.4.7 MDIO User Command Complete Interrupt (Unmasked) Register (USERINTRAW)

The MDIO user command complete interrupt (unmasked) register (USERINTRAW) is shown in [Figure 27-31](#) and described in [Table 27-31](#).

**Figure 27-31. MDIO User Command Complete Interrupt (Unmasked) Register (USERINTRAW)**



LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear (writing a 0 has no effect); -n = value after reset

**Table 27-31. MDIO User Command Complete Interrupt (Unmasked) Register (USERINTRAW) Field Descriptions**

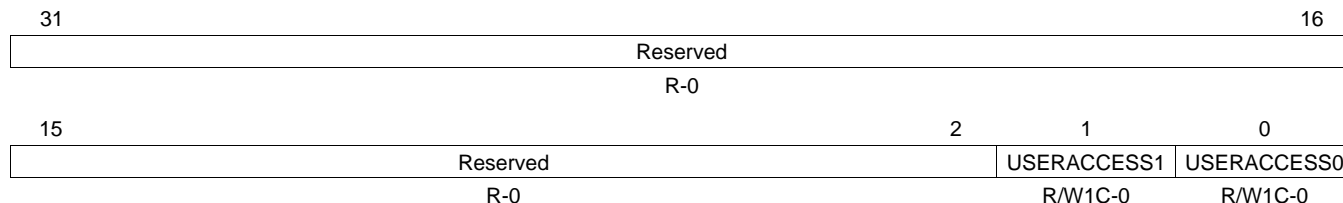
Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	USERACCESS1	0	MDIO User command complete event bit. When asserted, the bit indicates that the previously scheduled PHY read or write command using the USERACCESS1 register has completed. Writing a 1 will clear the event, writing a 0 has no effect.
		0	No MDIO user command complete event.
		1	The previously scheduled PHY read or write command using MDIO user access register USERACCESS1 has completed.
0	USERACCESS0	0	MDIO User command complete event bit. When asserted, the bit indicates that the previously scheduled PHY read or write command using the USERACCESS0 register has completed. Writing a 1 will clear the event, writing a 0 has no effect.
		0	No MDIO user command complete event.
		1	The previously scheduled PHY read or write command using MDIO user access register USERACCESS0 has completed.



### 27.4.8 MDIO User Command Complete Interrupt (Masked) Register (USERINTMASKED)

The MDIO user command complete interrupt (masked) register (USERINTMASKED) is shown in [Figure 27-32](#) and described in [Table 27-32](#).

**Figure 27-32. MDIO User Command Complete Interrupt (Masked) Register (USERINTMASKED)**



LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear (writing a 0 has no effect); -n = value after reset

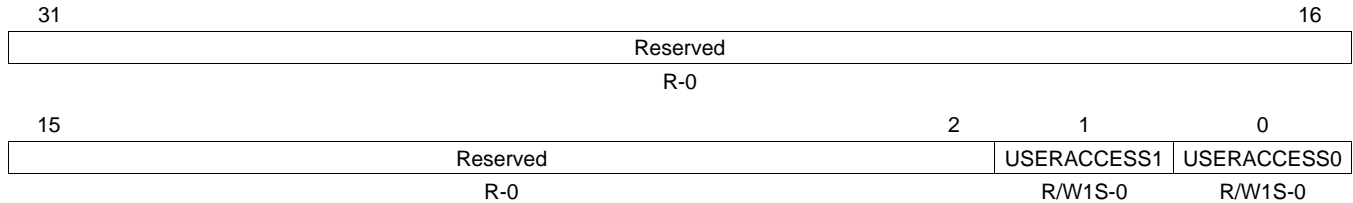
**Table 27-32. MDIO User Command Complete Interrupt (Masked) Register (USERINTMASKED) Field Descriptions**

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	USERACCESS1	0	Masked value of MDIO User command complete interrupt. When asserted, The bit indicates that the previously scheduled PHY read or write command using that particular USERACCESS1 register has completed. Writing a 1 will clear the interrupt, writing a 0 has no effect.
		1	No MDIO user command complete event.
		0	The previously scheduled PHY read or write command using MDIO user access register USERACCESS1 has completed and the corresponding bit in USERINTMASKSET is set to 1.
0	USERACCESS0	0	Masked value of MDIO User command complete interrupt. When asserted, The bit indicates that the previously scheduled PHY read or write command using that particular USERACCESS0 register has completed. Writing a 1 will clear the interrupt, writing a 0 has no effect.
		1	No MDIO user command complete event.
		0	The previously scheduled PHY read or write command using MDIO user access register USERACCESS0 has completed and the corresponding bit in USERINTMASKSET is set to 1.

### 27.4.9 MDIO User Command Complete Interrupt Mask Set Register (USERINTMASKSET)

The MDIO user command complete interrupt mask set register (USERINTMASKSET) is shown in [Figure 27-33](#) and described in [Table 27-33](#).

**Figure 27-33. MDIO User Command Complete Interrupt Mask Set Register (USERINTMASKSET)**



LEGEND: R/W = Read/Write; R = Read only; W1S = Write 1 to set (writing a 0 has no effect); -n = value after reset

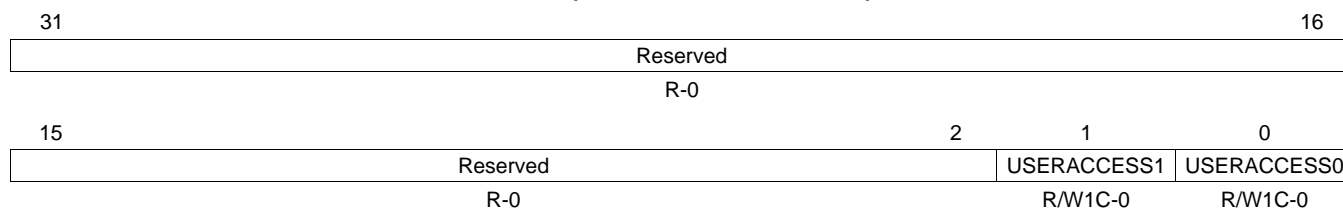
**Table 27-33. MDIO User Command Complete Interrupt Mask Set Register (USERINTMASKSET) Field Descriptions**

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	USERACCESS1	0	MDIO user interrupt mask set for USERINTMASKED[1]. Setting a bit to 1 will enable MDIO user command complete interrupts for the USERACCESS1 register. MDIO user interrupt for USERACCESS1 is disabled if the corresponding bit is 0. Writing a 0 to this bit has no effect.
		0	MDIO user command complete interrupts for the MDIO user access register USERACCESS0 is disabled.
		1	MDIO user command complete interrupts for the MDIO user access register USERACCESS0 is enabled.
0	USERACCESS0	0	MDIO user interrupt mask set for USERINTMASKED[0]. Setting a bit to 1 will enable MDIO user command complete interrupts for the USERACCESS0 register. MDIO user interrupt for USERACCESS0 is disabled if the corresponding bit is 0. Writing a 0 to this bit has no effect.
		0	MDIO user command complete interrupts for the MDIO user access register USERACCESS0 is disabled.
		1	MDIO user command complete interrupts for the MDIO user access register USERACCESS0 is enabled.

### 27.4.10 MDIO User Command Complete Interrupt Mask Clear Register (USERINTMASKCLEAR)

The MDIO user command complete interrupt mask clear register (USERINTMASKCLEAR) is shown in [Figure 27-34](#) and described in [Table 27-34](#).

**Figure 27-34. MDIO User Command Complete Interrupt Mask Clear Register (USERINTMASKCLEAR)**



LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear (writing a 0 has no effect); -n = value after reset

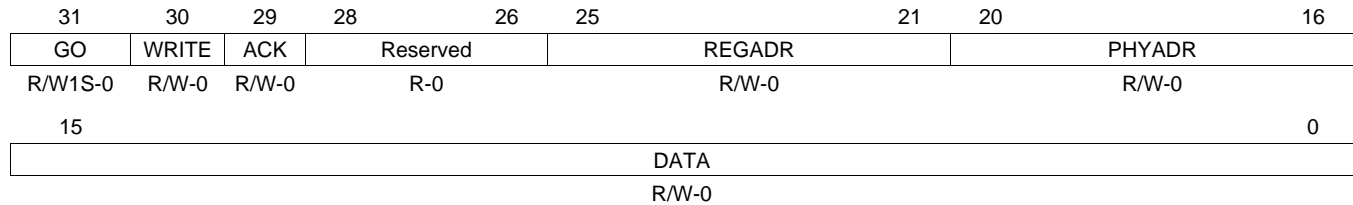
**Table 27-34. MDIO User Command Complete Interrupt Mask Clear Register (USERINTMASKCLEAR) Field Descriptions**

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	USERACCESS1	0	MDIO user command complete interrupt mask clear for USERINTMASKED[1]. Setting the bit to 1 will disable further user command complete interrupts for USERACCESS1. Writing a 0 to this bit has no effect.
		0	MDIO user command complete interrupts for the MDIO user access register USERACCESS1 is enabled.
		1	MDIO user command complete interrupts for the MDIO user access register USERACCESS1 is disabled.
0	USERACCESS0	0	MDIO user command complete interrupt mask clear for USERINTMASKED[0]. Setting the bit to 1 will disable further user command complete interrupts for USERACCESS0. Writing a 0 to this bit has no effect.
		0	MDIO user command complete interrupts for the MDIO user access register USERACCESS0 is enabled.
		1	MDIO user command complete interrupts for the MDIO user access register USERACCESS0 is disabled.

### 27.4.11 MDIO User Access Register 0 (USERACCESS0)

The MDIO user access register 0 (USERACCESS0) is shown in [Figure 27-35](#) and described in [Table 27-35](#).

**Figure 27-35. MDIO User Access Register 0 (USERACCESS0)**



LEGEND: R/W = Read/Write; R = Read only; W1S = Write 1 to set (writing a 0 has no effect); -n = value after reset

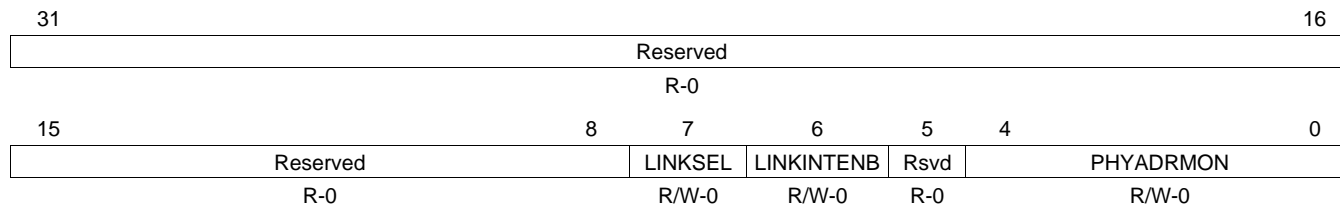
**Table 27-35. MDIO User Access Register 0 (USERACCESS0) Field Descriptions**

Bit	Field	Value	Description
31	GO	0-1	Go bit. Writing a 1 to this bit causes the MDIO state machine to perform an MDIO access when it is convenient for it to do so; this is not an instantaneous process. Writing a 0 to this bit has no effect. This bit is writeable only if the MDIO state machine is enabled. This bit will self clear when the requested access has been completed. Any writes to USERACCESS0 are blocked when the GO bit is 1.
30	WRITE	0 1	Write enable bit. Setting this bit to 1 causes the MDIO transaction to be a register write; otherwise, it is a register read. The user command is a read operation. The user command is a write operation.
29	ACK	0-1	Acknowledge bit. This bit is set if the PHY acknowledged the read transaction.
28-26	Reserved	0	Reserved
25-21	REGADR	0-1Fh	Register address bits. This field specifies the PHY register to be accessed for this transaction
20-16	PHYADR	0-1Fh	PHY address bits. This field specifies the PHY to be accessed for this transaction.
15-0	DATA	0-FFFFh	User data bits. These bits specify the data value read from or to be written to the specified PHY register.

### 27.4.12 MDIO User PHY Select Register 0 (USERPHYSEL0)

The MDIO user PHY select register 0 (USERPHYSEL0) is shown in [Figure 27-36](#) and described in [Table 27-36](#).

**Figure 27-36. MDIO User PHY Select Register 0 (USERPHYSEL0)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

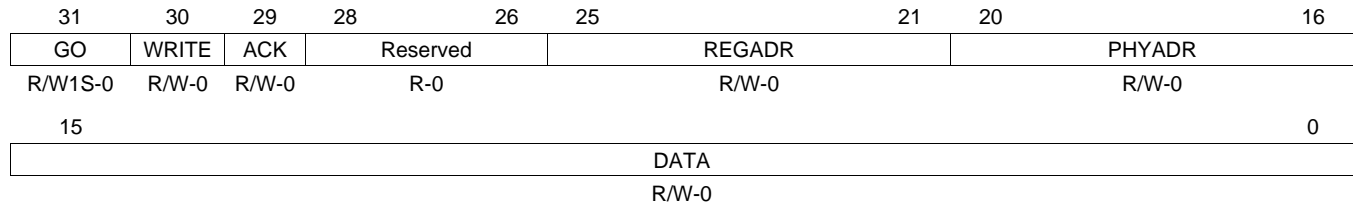
**Table 27-36. MDIO User PHY Select Register 0 (USERPHYSEL0) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	LINKSEL	0 1	Link status determination select bit. Default value is 0, which implies that the link status is determined by the MDIO state machine. This is the only option supported on this device. The link status is determined by the MDIO state machine. Not supported.
6	LINKINTENB	0 1	Link change interrupt enable. Set to 1 to enable link change status interrupts for PHY address specified in PHYADDRMON. Link change interrupts are disabled if this bit is cleared to 0. Link change interrupts are disabled. Link change status interrupts for PHY address specified in PHYADDRMON bits are enabled.
5	Reserved	0	Reserved
4-0	PHYADDRMON	0-1Fh	PHY address whose link status is to be monitored.

### 27.4.13 MDIO User Access Register 1 (USERACCESS1)

The MDIO user access register 1 (USERACCESS1) is shown in [Figure 27-37](#) and described in [Table 27-37](#).

**Figure 27-37. MDIO User Access Register 1 (USERACCESS1)**



LEGEND: R/W = Read/Write; R = Read only; W1S = Write 1 to set (writing a 0 has no effect); -n = value after reset

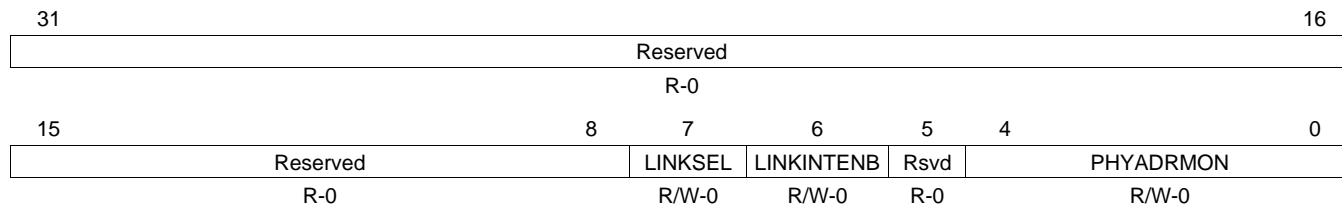
**Table 27-37. MDIO User Access Register 1 (USERACCESS1) Field Descriptions**

Bit	Field	Value	Description
31	GO	0-1	Go bit. Writing 1 to this bit causes the MDIO state machine to perform an MDIO access when it is convenient for it to do so; this is not an instantaneous process. Writing 0 to this bit has no effect. This bit is writeable only if the MDIO state machine is enabled. This bit will self clear when the requested access has been completed. Any writes to USERACCESS0 are blocked when the GO bit is 1.
30	WRITE	0 1	Write enable bit. Setting this bit to 1 causes the MDIO transaction to be a register write; otherwise, it is a register read. The user command is a read operation. The user command is a write operation.
29	ACK	0-1	Acknowledge bit. This bit is set if the PHY acknowledged the read transaction.
28-26	Reserved	0	Reserved
25-21	REGADR	0-1Fh	Register address bits. This field specifies the PHY register to be accessed for this transaction
20-16	PHYADR	0-1Fh	PHY address bits. This field specifies the PHY to be accessed for this transaction.
15-0	DATA	0-FFFFh	User data bits. These bits specify the data value read from or to be written to the specified PHY register.

### 27.4.14 MDIO User PHY Select Register 1 (USERPHYSEL1)

The MDIO user PHY select register 1 (USERPHYSEL1) is shown in [Figure 27-38](#) and described in [Table 27-38](#).

**Figure 27-38. MDIO User PHY Select Register 1 (USERPHYSEL1)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 27-38. MDIO User PHY Select Register 1 (USERPHYSEL1) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	LINKSEL	0 1	Link status determination select bit. Default value is 0, which implies that the link status is determined by the MDIO state machine. This is the only option supported on this device. The link status is determined by the MDIO state machine. Not supported.
6	LINKINTENB	0 1	Link change interrupt enable. Set to 1 to enable link change status interrupts for the PHY address specified in PHYADRMON. Link change interrupts are disabled if this bit is cleared to 0. Link change interrupts are disabled. Link change status interrupts for PHY address specified in PHYADRMON bits are enabled.
5	Reserved	0	PHY address whose link status is to be monitored.
4-0	PHYADRMON	0-1Fh	PHY address whose link status is to be monitored.

## 27.5 EMAC Module Registers

Table 27-39 lists the memory-mapped registers for the EMAC. The base address for these registers is 0xFCF78000.

**Table 27-39. Ethernet Media Access Controller (EMAC) Registers**

Offset	Acronym	Register Description	Section
0h	TXREVID	Transmit Revision ID Register	<a href="#">Section 27.5.1</a>
4h	TXCONTROL	Transmit Control Register	<a href="#">Section 27.5.2</a>
8h	TXTEARDOWN	Transmit Teardown Register	<a href="#">Section 27.5.3</a>
10h	RXREVID	Receive Revision ID Register	<a href="#">Section 27.5.4</a>
14h	RXCONTROL	Receive Control Register	<a href="#">Section 27.5.5</a>
18h	RXTEARDOWN	Receive Teardown Register	<a href="#">Section 27.5.6</a>
80h	TXINTSTATRAW	Transmit Interrupt Status (Unmasked) Register	<a href="#">Section 27.5.7</a>
84h	TXINTSTATMASKED	Transmit Interrupt Status (Masked) Register	<a href="#">Section 27.5.8</a>
88h	TXINTMASKSET	Transmit Interrupt Mask Set Register	<a href="#">Section 27.5.9</a>
8Ch	TXINTMASKCLEAR	Transmit Interrupt Clear Register	<a href="#">Section 27.5.10</a>
90h	MACINVECTOR	MAC Input Vector Register	<a href="#">Section 27.5.11</a>
94h	MACEOIVECTOR	MAC End Of Interrupt Vector Register	<a href="#">Section 27.5.12</a>
A0h	RXINTSTATRAW	Receive Interrupt Status (Unmasked) Register	<a href="#">Section 27.5.13</a>
A4h	RXINTSTATMASKED	Receive Interrupt Status (Masked) Register	<a href="#">Section 27.5.14</a>
A8h	RXINTMASKSET	Receive Interrupt Mask Set Register	<a href="#">Section 27.5.15</a>
ACh	RXINTMASKCLEAR	Receive Interrupt Mask Clear Register	<a href="#">Section 27.5.16</a>
B0h	MACINTSTATRAW	MAC Interrupt Status (Unmasked) Register	<a href="#">Section 27.5.17</a>
B4h	MACINTSTATMASKED	MAC Interrupt Status (Masked) Register	<a href="#">Section 27.5.18</a>
B8h	MACINTMASKSET	MAC Interrupt Mask Set Register	<a href="#">Section 27.5.19</a>
BCh	MACINTMASKCLEAR	MAC Interrupt Mask Clear Register	<a href="#">Section 27.5.20</a>
100h	RXMBPENABLE	Receive Multicast/Broadcast/Promiscuous Channel Enable Register	<a href="#">Section 27.5.21</a>
104h	RXUNICASTSET	Receive Unicast Enable Set Register	<a href="#">Section 27.5.22</a>
108h	RXUNICASTCLEAR	Receive Unicast Clear Register	<a href="#">Section 27.5.23</a>
10Ch	RXMAXLEN	Receive Maximum Length Register	<a href="#">Section 27.5.24</a>
110h	RXBUFFEROFFSET	Receive Buffer Offset Register	<a href="#">Section 27.5.25</a>
114h	RXFILTERLOWTHRESH	Receive Filter Low Priority Frame Threshold Register	<a href="#">Section 27.5.26</a>
120h	RX0FLOWTHRESH	Receive Channel 0 Flow Control Threshold Register	<a href="#">Section 27.5.27</a>
124h	RX1FLOWTHRESH	Receive Channel 1 Flow Control Threshold Register	<a href="#">Section 27.5.27</a>
128h	RX2FLOWTHRESH	Receive Channel 2 Flow Control Threshold Register	<a href="#">Section 27.5.27</a>
12Ch	RX3FLOWTHRESH	Receive Channel 3 Flow Control Threshold Register	<a href="#">Section 27.5.27</a>
130h	RX4FLOWTHRESH	Receive Channel 4 Flow Control Threshold Register	<a href="#">Section 27.5.27</a>
134h	RX5FLOWTHRESH	Receive Channel 5 Flow Control Threshold Register	<a href="#">Section 27.5.27</a>
138h	RX6FLOWTHRESH	Receive Channel 6 Flow Control Threshold Register	<a href="#">Section 27.5.27</a>
13Ch	RX7FLOWTHRESH	Receive Channel 7 Flow Control Threshold Register	<a href="#">Section 27.5.27</a>
140h	RX0FREEBUFFER	Receive Channel 0 Free Buffer Count Register	<a href="#">Section 27.5.28</a>
144h	RX1FREEBUFFER	Receive Channel 1 Free Buffer Count Register	<a href="#">Section 27.5.28</a>
148h	RX2FREEBUFFER	Receive Channel 2 Free Buffer Count Register	<a href="#">Section 27.5.28</a>
14Ch	RX3FREEBUFFER	Receive Channel 3 Free Buffer Count Register	<a href="#">Section 27.5.28</a>
150h	RX4FREEBUFFER	Receive Channel 4 Free Buffer Count Register	<a href="#">Section 27.5.28</a>
154h	RX5FREEBUFFER	Receive Channel 5 Free Buffer Count Register	<a href="#">Section 27.5.28</a>
158h	RX6FREEBUFFER	Receive Channel 6 Free Buffer Count Register	<a href="#">Section 27.5.28</a>
15Ch	RX7FREEBUFFER	Receive Channel 7 Free Buffer Count Register	<a href="#">Section 27.5.28</a>
160h	MACCONTROL	MAC Control Register	<a href="#">Section 27.5.29</a>



**Table 27-39. Ethernet Media Access Controller (EMAC) Registers (continued)**

Offset	Acronym	Register Description	Section
164h	MACSTATUS	MAC Status Register	<a href="#">Section 27.5.30</a>
168h	EMCONTROL	Emulation Control Register	<a href="#">Section 27.5.31</a>
16Ch	FIFOCONTROL	FIFO Control Register	<a href="#">Section 27.5.32</a>
170h	MACCONFIG	MAC Configuration Register	<a href="#">Section 27.5.33</a>
174h	SOFTRESET	Soft Reset Register	<a href="#">Section 27.5.34</a>
1D0h	MACSRCADDRLO	MAC Source Address Low Bytes Register	<a href="#">Section 27.5.35</a>
1D4h	MACSRCADDRHI	MAC Source Address High Bytes Register	<a href="#">Section 27.5.36</a>
1D8h	MACHASH1	MAC Hash Address Register 1	<a href="#">Section 27.5.37</a>
1DCh	MACHASH2	MAC Hash Address Register 2	<a href="#">Section 27.5.38</a>
1E0h	BOFFTEST	Back Off Test Register	<a href="#">Section 27.5.39</a>
1E4h	TPACETEST	Transmit Pacing Algorithm Test Register	<a href="#">Section 27.5.40</a>
1E8h	RXPAUSE	Receive Pause Timer Register	<a href="#">Section 27.5.41</a>
1ECh	TXPAUSE	Transmit Pause Timer Register	<a href="#">Section 27.5.42</a>
500h	MACADDRLO	MAC Address Low Bytes Register, Used in Receive Address Matching	<a href="#">Section 27.5.43</a>
504h	MACADDRHI	MAC Address High Bytes Register, Used in Receive Address Matching	<a href="#">Section 27.5.44</a>
508h	MACINDEX	MAC Index Register	<a href="#">Section 27.5.45</a>
600h	TX0HDP	Transmit Channel 0 DMA Head Descriptor Pointer Register	<a href="#">Section 27.5.46</a>
604h	TX1HDP	Transmit Channel 1 DMA Head Descriptor Pointer Register	<a href="#">Section 27.5.46</a>
608h	TX2HDP	Transmit Channel 2 DMA Head Descriptor Pointer Register	<a href="#">Section 27.5.46</a>
60Ch	TX3HDP	Transmit Channel 3 DMA Head Descriptor Pointer Register	<a href="#">Section 27.5.46</a>
610h	TX4HDP	Transmit Channel 4 DMA Head Descriptor Pointer Register	<a href="#">Section 27.5.46</a>
614h	TX5HDP	Transmit Channel 5 DMA Head Descriptor Pointer Register	<a href="#">Section 27.5.46</a>
618h	TX6HDP	Transmit Channel 6 DMA Head Descriptor Pointer Register	<a href="#">Section 27.5.46</a>
61Ch	TX7HDP	Transmit Channel 7 DMA Head Descriptor Pointer Register	<a href="#">Section 27.5.46</a>
620h	RX0HDP	Receive Channel 0 DMA Head Descriptor Pointer Register	<a href="#">Section 27.5.47</a>
624h	RX1HDP	Receive Channel 1 DMA Head Descriptor Pointer Register	<a href="#">Section 27.5.47</a>
628h	RX2HDP	Receive Channel 2 DMA Head Descriptor Pointer Register	<a href="#">Section 27.5.47</a>
62Ch	RX3HDP	Receive Channel 3 DMA Head Descriptor Pointer Register	<a href="#">Section 27.5.47</a>
630h	RX4HDP	Receive Channel 4 DMA Head Descriptor Pointer Register	<a href="#">Section 27.5.47</a>
634h	RX5HDP	Receive Channel 5 DMA Head Descriptor Pointer Register	<a href="#">Section 27.5.47</a>
638h	RX6HDP	Receive Channel 6 DMA Head Descriptor Pointer Register	<a href="#">Section 27.5.47</a>
63Ch	RX7HDP	Receive Channel 7 DMA Head Descriptor Pointer Register	<a href="#">Section 27.5.47</a>
640h	TX0CP	Transmit Channel 0 Completion Pointer Register	<a href="#">Section 27.5.48</a>
644h	TX1CP	Transmit Channel 1 Completion Pointer Register	<a href="#">Section 27.5.48</a>
648h	TX2CP	Transmit Channel 2 Completion Pointer Register	<a href="#">Section 27.5.48</a>
64Ch	TX3CP	Transmit Channel 3 Completion Pointer Register	<a href="#">Section 27.5.48</a>
650h	TX4CP	Transmit Channel 4 Completion Pointer Register	<a href="#">Section 27.5.48</a>
654h	TX5CP	Transmit Channel 5 Completion Pointer Register	<a href="#">Section 27.5.48</a>
658h	TX6CP	Transmit Channel 6 Completion Pointer Register	<a href="#">Section 27.5.48</a>
65Ch	TX7CP	Transmit Channel 7 Completion Pointer Register	<a href="#">Section 27.5.48</a>
660h	RX0CP	Receive Channel 0 Completion Pointer Register	<a href="#">Section 27.5.49</a>
664h	RX1CP	Receive Channel 1 Completion Pointer Register	<a href="#">Section 27.5.49</a>
668h	RX2CP	Receive Channel 2 Completion Pointer Register	<a href="#">Section 27.5.49</a>
66Ch	RX3CP	Receive Channel 3 Completion Pointer Register	<a href="#">Section 27.5.49</a>
670h	RX4CP	Receive Channel 4 Completion Pointer Register	<a href="#">Section 27.5.49</a>
674h	RX5CP	Receive Channel 5 Completion Pointer Register	<a href="#">Section 27.5.49</a>
678h	RX6CP	Receive Channel 6 Completion Pointer Register	<a href="#">Section 27.5.49</a>

**Table 27-39. Ethernet Media Access Controller (EMAC) Registers (continued)**

Offset	Acronym	Register Description	Section
67Ch	RX7CP	Receive Channel 7 Completion Pointer Register	<a href="#">Section 27.5.49</a>
<b>Network Statistics Registers</b>			
200h	RXGOODFRAMES	Good Receive Frames Register	<a href="#">Section 27.5.50.1</a>
204h	RXBCASTFRAMES	Broadcast Receive Frames Register	<a href="#">Section 27.5.50.2</a>
208h	RXMCASTFRAMES	Multicast Receive Frames Register	<a href="#">Section 27.5.50.3</a>
20Ch	RXPAUSEFRAMES	Pause Receive Frames Register	<a href="#">Section 27.5.50.4</a>
210h	RXCRCERRORS	Receive CRC Errors Register	<a href="#">Section 27.5.50.5</a>
214h	RXALIGNCODEERRORS	Receive Alignment/Code Errors Register	<a href="#">Section 27.5.50.6</a>
218h	RXOVERSIZED	Receive Oversized Frames Register	<a href="#">Section 27.5.50.7</a>
21Ch	RXJABBER	Receive Jabber Frames Register	<a href="#">Section 27.5.50.8</a>
220h	RXUNDERSIZED	Receive Undersized Frames Register	<a href="#">Section 27.5.50.9</a>
224h	RXFRAGMENTS	Receive Frame Fragments Register	<a href="#">Section 27.5.50.10</a>
228h	RXFILTERED	Filtered Receive Frames Register	<a href="#">Section 27.5.50.11</a>
22Ch	RXQOSFILTERED	Receive QOS Filtered Frames Register	<a href="#">Section 27.5.50.12</a>
230h	RXOCTETS	Receive Octet Frames Register	<a href="#">Section 27.5.50.13</a>
234h	TXGOODFRAMES	Good Transmit Frames Register	<a href="#">Section 27.5.50.14</a>
238h	TXBCASTFRAMES	Broadcast Transmit Frames Register	<a href="#">Section 27.5.50.15</a>
23Ch	TXMCASTFRAMES	Multicast Transmit Frames Register	<a href="#">Section 27.5.50.16</a>
240h	TXPAUSEFRAMES	Pause Transmit Frames Register	<a href="#">Section 27.5.50.17</a>
244h	TXDEFERRED	Deferred Transmit Frames Register	<a href="#">Section 27.5.50.18</a>
248h	TXCOLLISION	Transmit Collision Frames Register	<a href="#">Section 27.5.50.19</a>
24Ch	TXSINGLECOLL	Transmit Single Collision Frames Register	<a href="#">Section 27.5.50.20</a>
250h	TXMULTICOLL	Transmit Multiple Collision Frames Register	<a href="#">Section 27.5.50.21</a>
254h	TXEXCESSIVECOLL	Transmit Excessive Collision Frames Register	<a href="#">Section 27.5.50.22</a>
258h	TXLATECOLL	Transmit Late Collision Frames Register	<a href="#">Section 27.5.50.23</a>
25Ch	TXUNDERRUN	Transmit Underrun Error Register	<a href="#">Section 27.5.50.24</a>
260h	TXCARRIERSENSE	Transmit Carrier Sense Errors Register	<a href="#">Section 27.5.50.25</a>
264h	TXOCTETS	Transmit Octet Frames Register	<a href="#">Section 27.5.50.26</a>
268h	FRAME64	Transmit and Receive 64 Octet Frames Register	<a href="#">Section 27.5.50.27</a>
26Ch	FRAME65T127	Transmit and Receive 65 to 127 Octet Frames Register	<a href="#">Section 27.5.50.28</a>
270h	FRAME128T255	Transmit and Receive 128 to 255 Octet Frames Register	<a href="#">Section 27.5.50.29</a>
274h	FRAME256T511	Transmit and Receive 256 to 511 Octet Frames Register	<a href="#">Section 27.5.50.30</a>

**Table 27-39. Ethernet Media Access Controller (EMAC) Registers (continued)**

Offset	Acronym	Register Description	Section
278h	FRAME512T1023	Transmit and Receive 512 to 1023 Octet Frames Register	<a href="#">Section 27.5.50.31</a>
27Ch	FRAME1024TUP	Transmit and Receive 1024 to RXMAXLEN Octet Frames Register	<a href="#">Section 27.5.50.32</a>
280h	NETOCTETS	Network Octet Frames Register	<a href="#">Section 27.5.50.33</a>
284h	RXSOFOVERRUNS	Receive FIFO or DMA Start of Frame Overruns Register	<a href="#">Section 27.5.50.34</a>
288h	RXMOFOVERRUNS	Receive FIFO or DMA Middle of Frame Overruns Register	<a href="#">Section 27.5.50.35</a>
28Ch	RXDMAOVERRUNS	Receive DMA Overruns Register	<a href="#">Section 27.5.50.36</a>

### 27.5.1 Transmit Revision ID Register (TXREVID)

The transmit revision ID register (TXREVID) is shown in [Figure 27-39](#) and described in [Table 27-40](#).

**Figure 27-39. Transmit Revision ID Register (TXREVID)**



LEGEND: R = Read only; -n = value after reset

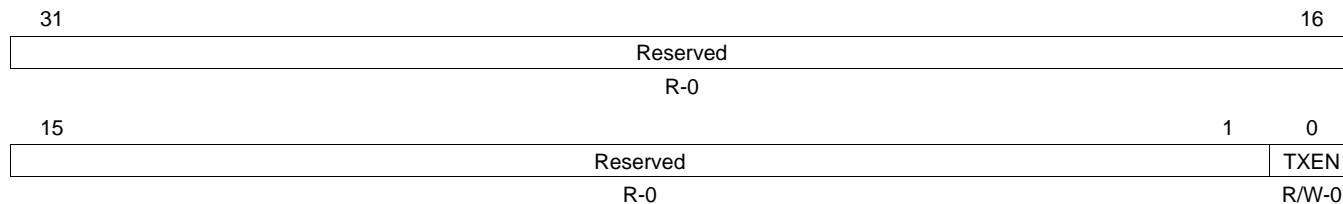
**Table 27-40. Transmit Revision ID Register (TXREVID) Field Descriptions**

Bit	Field	Value	Description
31-0	TXREV	4EC0 020Dh	Transmit module revision Current transmit revision value

### 27.5.2 Transmit Control Register (TXCONTROL)

The transmit control register (TXCONTROL) is shown in [Figure 27-40](#) and described in [Table 27-41](#).

**Figure 27-40. Transmit Control Register (TXCONTROL)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

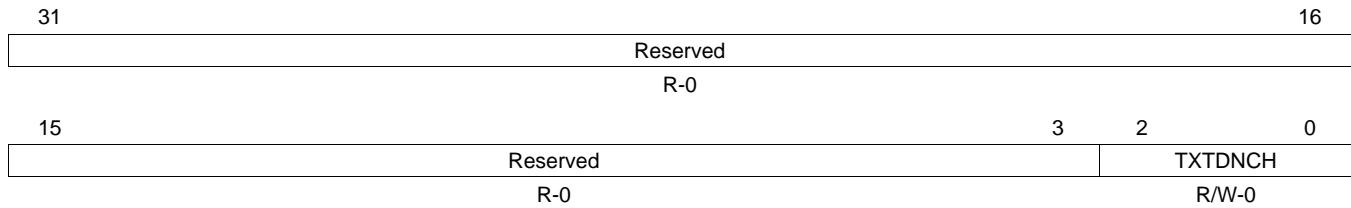
**Table 27-41. Transmit Control Register (TXCONTROL) Field Descriptions**

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	TXEN	0	Transmit enable Transmit is disabled.
		1	Transmit is enabled.

### 27.5.3 Transmit Teardown Register (TXTEARDOWN)

The transmit teardown register (TXTEARDOWN) is shown in [Figure 27-41](#) and described in [Table 27-42](#).

**Figure 27-41. Transmit Teardown Register (TXTEARDOWN)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

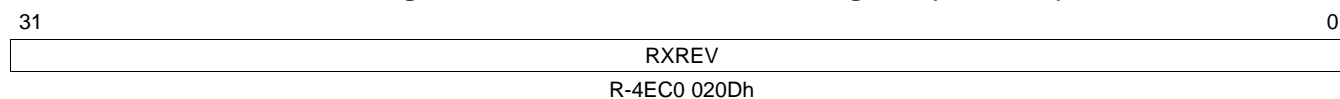
**Table 27-42. Transmit Teardown Register (TXTEARDOWN) Field Descriptions**

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2-0	TXTDNCH	0-7h	Transmit teardown channel. The transmit channel teardown is commanded by writing the encoded value of the transmit channel to be torn down. The teardown register is read as 0.
		0	Teardown transmit channel 0
		1h	Teardown transmit channel 1
		2h	Teardown transmit channel 2
		3h	Teardown transmit channel 3
		4h	Teardown transmit channel 4
		5h	Teardown transmit channel 5
		6h	Teardown transmit channel 6
		7h	Teardown transmit channel 7

### 27.5.4 Receive Revision ID Register (RXREVID)

The receive revision ID register (RXREVID) is shown in [Figure 27-42](#) and described in [Table 27-43](#).

**Figure 27-42. Receive Revision ID Register (RXREVID)**



LEGEND: R = Read only; -n = value after reset

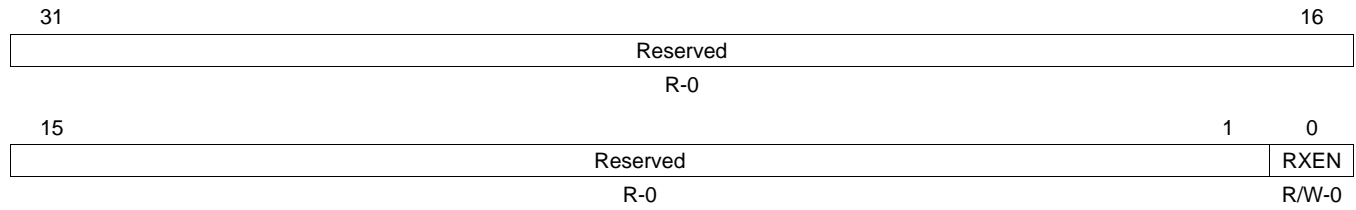
**Table 27-43. Receive Revision ID Register (RXREVID) Field Descriptions**

Bit	Field	Value	Description
31-0	RXREV	4EC0 020Dh	Receive module revision Current receive revision value

### 27.5.5 Receive Control Register (RXCONTROL)

The receive control register (RXCONTROL) is shown in [Figure 27-43](#) and described in [Table 27-44](#).

**Figure 27-43. Receive Control Register (RXCONTROL)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 27-44. Receive Control Register (RXCONTROL) Field Descriptions**

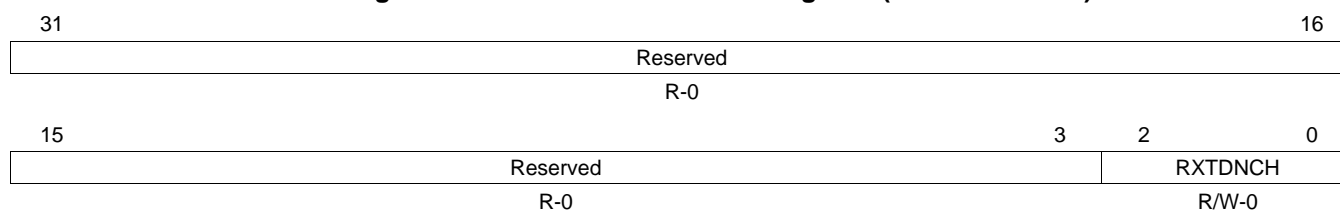
Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	RXEN	0	Receive enable Receive is disabled.
		1	Receive is enabled.



### 27.5.6 Receive Teardown Register (RXTEARDOWN)

The receive teardown register (RXTEARDOWN) is shown in [Figure 27-44](#) and described in [Table 27-45](#).

**Figure 27-44. Receive Teardown Register (RXTEARDOWN)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

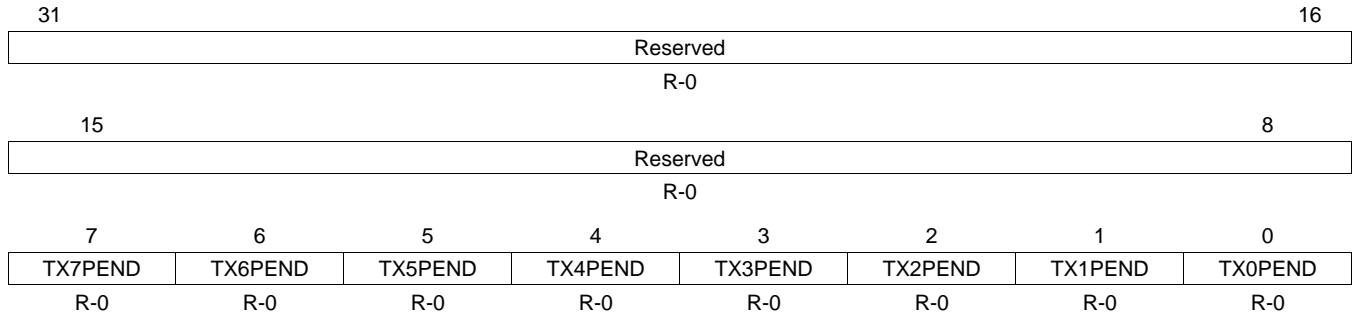
**Table 27-45. Receive Teardown Register (RXTEARDOWN) Field Descriptions**

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2-0	RXTDNCH	0-7h	Receive teardown channel. The receive channel teardown is commanded by writing the encoded value of the receive channel to be torn down. The teardown register is read as 0.
		0	Teardown receive channel 0
		1h	Teardown receive channel 1
		2h	Teardown receive channel 2
		3h	Teardown receive channel 3
		4h	Teardown receive channel 4
		5h	Teardown receive channel 5
		6h	Teardown receive channel 6
		7h	Teardown receive channel 7

**27.5.7 Transmit Interrupt Status (Unmasked) Register (TXINTSTATRAW)**

The transmit interrupt status (unmasked) register (TXINTSTATRAW) is shown in [Figure 27-45](#) and described in [Table 27-46](#).

**Figure 27-45. Transmit Interrupt Status (Unmasked) Register (TXINTSTATRAW)**



LEGEND: R = Read only; -n = value after reset

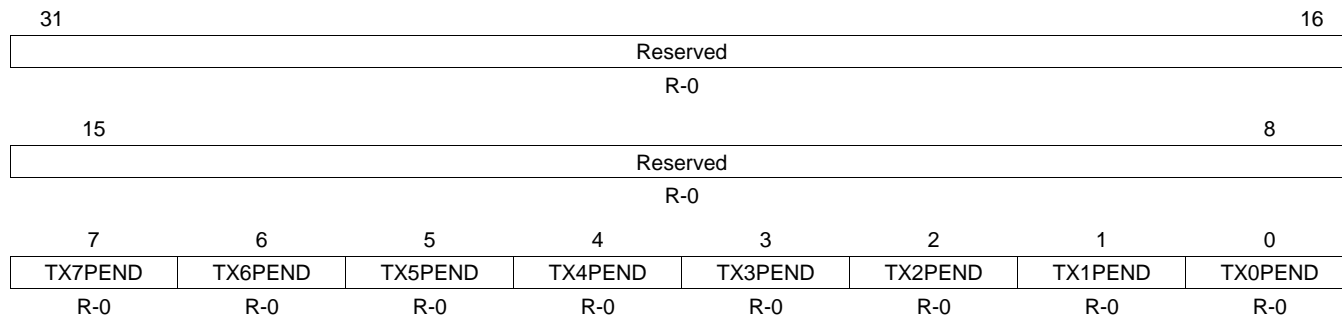
**Table 27-46. Transmit Interrupt Status (Unmasked) Register (TXINTSTATRAW) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	TX7PEND	0-1	TX7PEND raw interrupt read (before mask)
6	TX6PEND	0-1	TX6PEND raw interrupt read (before mask)
5	TX5PEND	0-1	TX5PEND raw interrupt read (before mask)
4	TX4PEND	0-1	TX4PEND raw interrupt read (before mask)
3	TX3PEND	0-1	TX3PEND raw interrupt read (before mask)
2	TX2PEND	0-1	TX2PEND raw interrupt read (before mask)
1	TX1PEND	0-1	TX1PEND raw interrupt read (before mask)
0	TX0PEND	0-1	TX0PEND raw interrupt read (before mask)

### 27.5.8 Transmit Interrupt Status (Masked) Register (TXINTSTATMASKED)

The transmit interrupt status (masked) register (TXINTSTATMASKED) is shown in [Figure 27-46](#) and described in [Table 27-47](#).

**Figure 27-46. Transmit Interrupt Status (Masked) Register (TXINTSTATMASKED)**



LEGEND: R = Read only; -n = value after reset

**Table 27-47. Transmit Interrupt Status (Masked) Register (TXINTSTATMASKED) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	TX7PEND	0-1	TX7PEND masked interrupt read
6	TX6PEND	0-1	TX6PEND masked interrupt read
5	TX5PEND	0-1	TX5PEND masked interrupt read
4	TX4PEND	0-1	TX4PEND masked interrupt read
3	TX3PEND	0-1	TX3PEND masked interrupt read
2	TX2PEND	0-1	TX2PEND masked interrupt read
1	TX1PEND	0-1	TX1PEND masked interrupt read
0	TX0PEND	0-1	TX0PEND masked interrupt read





### 27.5.11 MAC Input Vector Register (MACINVECTOR)

The MAC input vector register (MACINVECTOR) is shown in [Figure 27-49](#) and described in [Table 27-50](#).

**Figure 27-49. MAC Input Vector Register (MACINVECTOR)**

31	28	27	26	25	24	23	16	
Reserved		STATPEND	HOSTPEND	LINKINT0	USERINT0	TXPEND		
R-0		R-0	R-0	R-0	R-0	R-0		
15						8	7	0
RXTHRESHPEND						RXPEND		
R-0						R-0		

LEGEND: R = Read only; -n = value after reset

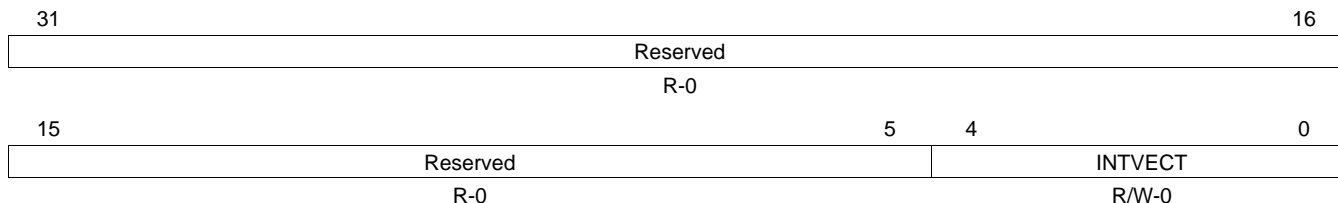
**Table 27-50. MAC Input Vector Register (MACINVECTOR) Field Descriptions**

Bit	Field	Value	Description
31-28	Reserved	0	Reserved
27	STATPEND	0-1	EMAC module statistics interrupt (STATPEND) pending status bit
26	HOSTPEND	0-1	EMAC module host error interrupt (HOSTPEND) pending status bit
25	LINKINT0	0-1	MDIO module USERPHYSEL0 (LINKINT0) status bit
24	USERINT0	0-1	MDIO module USERACCESS0 (USERINT0) status bit
23-16	TXPEND	0-FFh	Transmit channels 0-7 interrupt (TXnPEND) pending status. Bit 16 is TX0PEND.
15-8	RXTHRESHPEND	0-FFh	Receive channels 0-7 interrupt (RXnTHRESHPEND) pending status. Bit 8 is RX0THRESHPEND.
7-0	RXPEND	0-FFh	Receive channels 0-7 interrupt (RXnPEND) pending status bit. Bit 0 is RX0PEND.

### 27.5.12 MAC End Of Interrupt Vector Register (MACEOIVECTOR)

The MAC end of interrupt vector register (MACEOIVECTOR) is shown in [Figure 27-50](#) and described in [Table 27-51](#).

**Figure 27-50. MAC End Of Interrupt Vector Register (MACEOIVECTOR)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

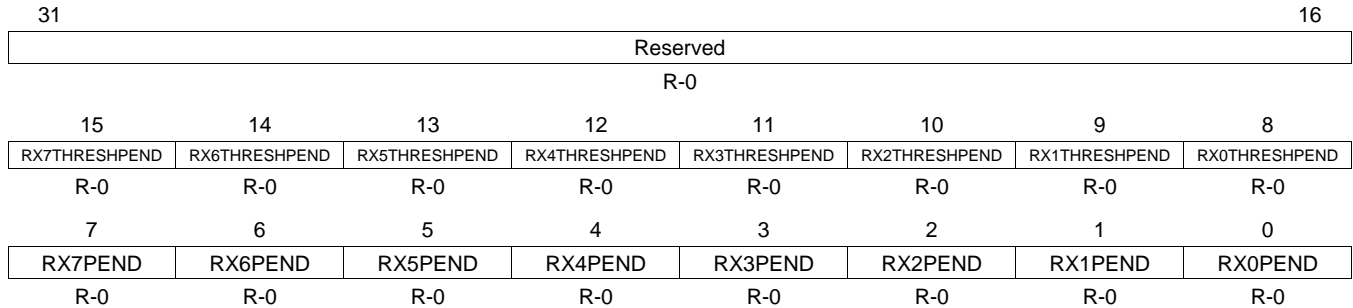
**Table 27-51. MAC End Of Interrupt Vector Register (MACEOIVECTOR) Field Descriptions**

Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4-0	INTVECT	0-1Fh	Acknowledge EMAC Control Module Interrupts
		0h	Acknowledge C0RXTHRESH Interrupt
		1h	Acknowledge C0RX Interrupt
		2h	Acknowledge C0TX Interrupt
		3h	Acknowledge C0MISC Interrupt (STATPEND, HOSTPEND, MDIO LINKINT0, MDIO USERINT0)
		4h	Acknowledge C1RXTHRESH Interrupt
		5h	Acknowledge C1RX Interrupt
		6h	Acknowledge C1TX Interrupt
		7h	Acknowledge C1MISC Interrupt (STATPEND, HOSTPEND, MDIO LINKINT0, MDIO USERINT0)
		8h	Acknowledge C2RXTHRESH Interrupt
		9h	Acknowledge C2RX Interrupt
		Ah	Acknowledge C2TX Interrupt
		Bh	Acknowledge C2MISC Interrupt (STATPEND, HOSTPEND, MDIO LINKINT0, MDIO USERINT0)
		Ch-1Fh	Reserved

### 27.5.13 Receive Interrupt Status (Unmasked) Register (RXINTSTATRAW)

The receive interrupt status (unmasked) register (RXINTSTATRAW) is shown in [Figure 27-51](#) and described in [Table 27-52](#).

**Figure 27-51. Receive Interrupt Status (Unmasked) Register (RXINTSTATRAW)**



LEGEND: R = Read only; -n = value after reset

**Table 27-52. Receive Interrupt Status (Unmasked) Register (RXINTSTATRAW) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	RX7THRESHPEND	0-1	RX7THRESHPEND raw interrupt read (before mask)
14	RX6THRESHPEND	0-1	RX6THRESHPEND raw interrupt read (before mask)
13	RX5THRESHPEND	0-1	RX5THRESHPEND raw interrupt read (before mask)
12	RX4THRESHPEND	0-1	RX4THRESHPEND raw interrupt read (before mask)
11	RX3THRESHPEND	0-1	RX3THRESHPEND raw interrupt read (before mask)
10	RX2THRESHPEND	0-1	RX2THRESHPEND raw interrupt read (before mask)
9	RX1THRESHPEND	0-1	RX1THRESHPEND raw interrupt read (before mask)
8	RX0THRESHPEND	0-1	RX0THRESHPEND raw interrupt read (before mask)
7	RX7PEND	0-1	RX7PEND raw interrupt read (before mask)
6	RX6PEND	0-1	RX6PEND raw interrupt read (before mask)
5	RX5PEND	0-1	RX5PEND raw interrupt read (before mask)
4	RX4PEND	0-1	RX4PEND raw interrupt read (before mask)
3	RX3PEND	0-1	RX3PEND raw interrupt read (before mask)
2	RX2PEND	0-1	RX2PEND raw interrupt read (before mask)
1	RX1PEND	0-1	RX1PEND raw interrupt read (before mask)
0	RX0PEND	0-1	RX0PEND raw interrupt read (before mask)



### 27.5.14 Receive Interrupt Status (Masked) Register (RXINTSTATMASKED)

The receive interrupt status (masked) register (RXINTSTATMASKED) is shown in [Figure 27-52](#) and described in [Table 27-53](#).

**Figure 27-52. Receive Interrupt Status (Masked) Register (RXINTSTATMASKED)**

Reserved							
R-0							
15	14	13	12	11	10	9	8
RX7THRESHPEND	RX6THRESHPEND	RX5THRESHPEND	RX4THRESHPEND	RX3THRESHPEND	RX2THRESHPEND	RX1THRESHPEND	RX0THRESHPEND
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
RX7PEND	RX6PEND	RX5PEND	RX4PEND	RX3PEND	RX2PEND	RX1PEND	RX0PEND
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; -n = value after reset

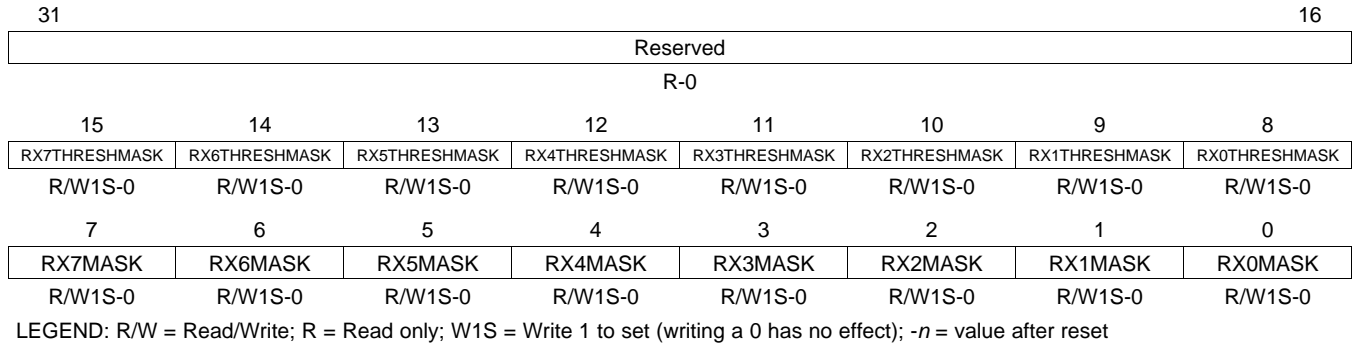
**Table 27-53. Receive Interrupt Status (Masked) Register (RXINTSTATMASKED) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	RX7THRESHPEND	0-1	RX7THRESHPEND masked interrupt read
14	RX6THRESHPEND	0-1	RX6THRESHPEND masked interrupt read
13	RX5THRESHPEND	0-1	RX5THRESHPEND masked interrupt read
12	RX4THRESHPEND	0-1	RX4THRESHPEND masked interrupt read
11	RX3THRESHPEND	0-1	RX3THRESHPEND masked interrupt read
10	RX2THRESHPEND	0-1	RX2THRESHPEND masked interrupt read
9	RX1THRESHPEND	0-1	RX1THRESHPEND masked interrupt read
8	RX0THRESHPEND	0-1	RX0THRESHPEND masked interrupt read
7	RX7PEND	0-1	RX7PEND masked interrupt read
6	RX6PEND	0-1	RX6PEND masked interrupt read
5	RX5PEND	0-1	RX5PEND masked interrupt read
4	RX4PEND	0-1	RX4PEND masked interrupt read
3	RX3PEND	0-1	RX3PEND masked interrupt read
2	RX2PEND	0-1	RX2PEND masked interrupt read
1	RX1PEND	0-1	RX1PEND masked interrupt read
0	RX0PEND	0-1	RX0PEND masked interrupt read

### 27.5.15 Receive Interrupt Mask Set Register (RXINTMASKSET)

The receive interrupt mask set register (RXINTMASKSET) is shown in [Figure 27-53](#) and described in [Table 27-54](#).

**Figure 27-53. Receive Interrupt Mask Set Register (RXINTMASKSET)**



**Table 27-54. Receive Interrupt Mask Set Register (RXINTMASKSET) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	RX7THRESHMASK	0-1	Receive channel 7 threshold mask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
14	RX6THRESHMASK	0-1	Receive channel 6 threshold mask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
13	RX5THRESHMASK	0-1	Receive channel 5 threshold mask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
12	RX4THRESHMASK	0-1	Receive channel 4 threshold mask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
11	RX3THRESHMASK	0-1	Receive channel 3 threshold mask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
10	RX2THRESHMASK	0-1	Receive channel 2 threshold mask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
9	RX1THRESHMASK	0-1	Receive channel 1 threshold mask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
8	RX0THRESHMASK	0-1	Receive channel 0 threshold mask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
7	RX7MASK	0-1	Receive channel 7 mask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
6	RX6MASK	0-1	Receive channel 6 mask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
5	RX5MASK	0-1	Receive channel 5 mask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
4	RX4MASK	0-1	Receive channel 4 mask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
3	RX3MASK	0-1	Receive channel 3 mask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
2	RX2MASK	0-1	Receive channel 2 mask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
1	RX1MASK	0-1	Receive channel 1 mask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
0	RX0MASK	0-1	Receive channel 0 mask set bit. Write 1 to enable interrupt; a write of 0 has no effect.

### 27.5.16 Receive Interrupt Mask Clear Register (RXINTMASKCLEAR)

The receive interrupt mask clear register (RXINTMASKCLEAR) is shown in [Figure 27-54](#) and described in [Table 27-55](#).

**Figure 27-54. Receive Interrupt Mask Clear Register (RXINTMASKCLEAR)**

Reserved							
R-0							
15	14	13	12	11	10	9	8
RX7THRESHMASK	RX6THRESHMASK	RX5THRESHMASK	RX4THRESHMASK	RX3THRESHMASK	RX2THRESHMASK	RX1THRESHMASK	RX0THRESHMASK
R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0
7	6	5	4	3	2	1	0
RX7MASK	RX6MASK	RX5MASK	RX4MASK	RX3MASK	RX2MASK	RX1MASK	RX0MASK
R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear (writing a 0 has no effect); -n = value after reset

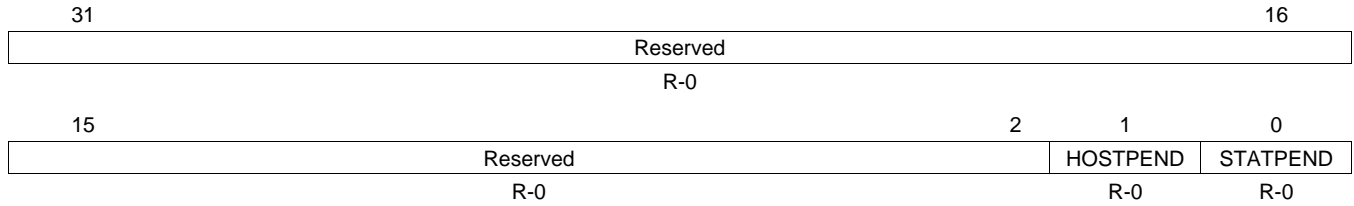
**Table 27-55. Receive Interrupt Mask Clear Register (RXINTMASKCLEAR) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	RX7THRESHMASK	0-1	Receive channel 7 threshold mask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
14	RX6THRESHMASK	0-1	Receive channel 6 threshold mask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
13	RX5THRESHMASK	0-1	Receive channel 5 threshold mask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
12	RX4THRESHMASK	0-1	Receive channel 4 threshold mask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
11	RX3THRESHMASK	0-1	Receive channel 3 threshold mask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
10	RX2THRESHMASK	0-1	Receive channel 2 threshold mask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
9	RX1THRESHMASK	0-1	Receive channel 1 threshold mask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
8	RX0THRESHMASK	0-1	Receive channel 0 threshold mask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
7	RX7MASK	0-1	Receive channel 7 mask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
6	RX6MASK	0-1	Receive channel 6 mask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
5	RX5MASK	0-1	Receive channel 5 mask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
4	RX4MASK	0-1	Receive channel 4 mask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
3	RX3MASK	0-1	Receive channel 3 mask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
2	RX2MASK	0-1	Receive channel 2 mask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
1	RX1MASK	0-1	Receive channel 1 mask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
0	RX0MASK	0-1	Receive channel 0 mask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.

**27.5.17 MAC Interrupt Status (Unmasked) Register (MACINTSTATRAW)**

The MAC interrupt status (unmasked) register (MACINTSTATRAW) is shown in [Figure 27-55](#) and described in [Table 27-56](#).

**Figure 27-55. MAC Interrupt Status (Unmasked) Register (MACINTSTATRAW)**



LEGEND: R = Read only; -n = value after reset

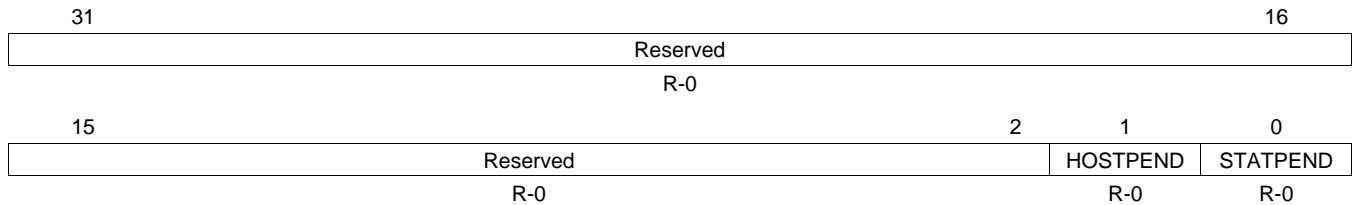
**Table 27-56. MAC Interrupt Status (Unmasked) Register (MACINTSTATRAW) Field Descriptions**

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	HOSTPEND	0-1	Host pending interrupt (HOSTPEND); raw interrupt read (before mask).
0	STATPEND	0-1	Statistics pending interrupt (STATPEND); raw interrupt read (before mask).

### 27.5.18 MAC Interrupt Status (Masked) Register (MACINTSTATMASKED)

The MAC interrupt status (masked) register (MACINTSTATMASKED) is shown in [Figure 27-56](#) and described in [Table 27-57](#).

**Figure 27-56. MAC Interrupt Status (Masked) Register (MACINTSTATMASKED)**



LEGEND: R = Read only; -n = value after reset

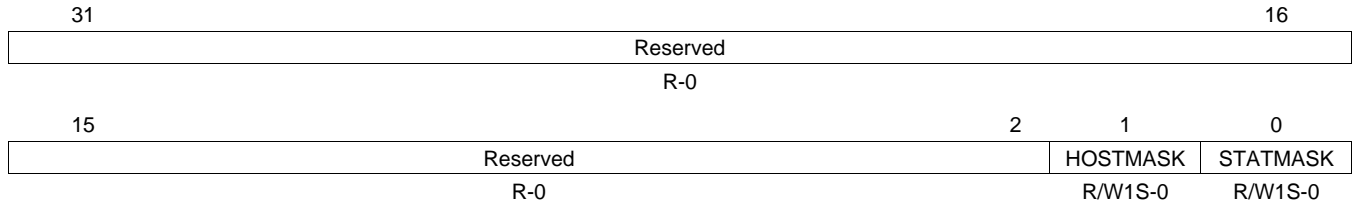
**Table 27-57. MAC Interrupt Status (Masked) Register (MACINTSTATMASKED) Field Descriptions**

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	HOSTPEND	0-1	Host pending interrupt (HOSTPEND); masked interrupt read.
0	STATPEND	0-1	Statistics pending interrupt (STATPEND); masked interrupt read.

**27.5.19 MAC Interrupt Mask Set Register (MACINTMASKSET)**

The MAC interrupt mask set register (MACINTMASKSET) is shown in [Figure 27-57](#) and described in [Table 27-58](#).

**Figure 27-57. MAC Interrupt Mask Set Register (MACINTMASKSET)**



LEGEND: R/W = Read/Write; R = Read only; W1S = Write 1 to set (writing a 0 has no effect); -n = value after reset

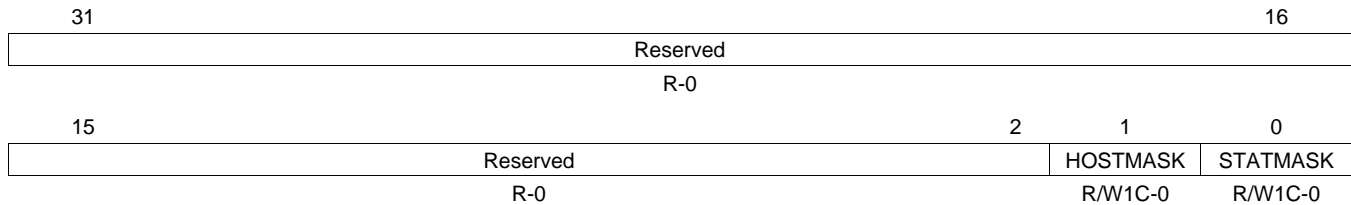
**Table 27-58. MAC Interrupt Mask Set Register (MACINTMASKSET) Field Descriptions**

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	HOSTMASK	0-1	Host error interrupt mask set bit. Write 1 to enable interrupt, a write of 0 has no effect.
0	STATMASK	0-1	Statistics interrupt mask set bit. Write 1 to enable interrupt, a write of 0 has no effect.

### 27.5.20 MAC Interrupt Mask Clear Register (MACINTMASKCLEAR)

The MAC interrupt mask clear register (MACINTMASKCLEAR) is shown in [Figure 27-58](#) and described in [Table 27-59](#).

**Figure 27-58. MAC Interrupt Mask Clear Register (MACINTMASKCLEAR)**



LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear (writing a 0 has no effect); -n = value after reset

**Table 27-59. MAC Interrupt Mask Clear Register (MACINTMASKCLEAR) Field Descriptions**

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	HOSTMASK	0-1	Host error interrupt mask clear bit. Write 1 to disable interrupt, a write of 0 has no effect.
0	STATMASK	0-1	Statistics interrupt mask clear bit. Write 1 to disable interrupt, a write of 0 has no effect.

**27.5.21 Receive Multicast/Broadcast/Promiscuous Channel Enable Register (RXMBPENABLE)**

The receive multicast/broadcast/promiscuous channel enable register (RXMBPENABLE) is shown in [Figure 27-59](#) and described in [Table 27-60](#).

**Figure 27-59. Receive Multicast/Broadcast/Promiscuous Channel Enable Register (RXMBPENABLE)**

31	30	29	28	27	25	24
Reserved	RXPASSCRC	RXQOSEN	RXNOCHAIN	Reserved		RXCMFEN
R-0	R/W-0	R/W-0	R/W-0	R-0		R/W-0
23	22	21	20	19	18	16
RXCSFEN	RXCEFEN	RXCAFEN	Reserved		RXPROMCH	
R/W-0	R/W-0	R/W-0	R-0		R/W-0	
15	14	13	12	11	10	8
Reserved		RXBROADEN	Reserved		RXBROADCH	
R-0		R/W-0	R-0		R/W-0	
7	6	5	4	3	2	0
Reserved		RXMULTEN	Reserved		RXMULTCH	
R-0		R/W-0	R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 27-60. Receive Multicast/Broadcast/Promiscuous Channel Enable Register (RXMBPENABLE) Field Descriptions**

Bit	Field	Value	Description
31	Reserved	0	Reserved
30	RXPASSCRC	0	Pass receive CRC enable bit
		0	Received CRC is discarded for all channels and is not included in the buffer descriptor packet length field.
		1	Received CRC is transferred to memory for all channels and is included in the buffer descriptor packet length.
29	RXQOSEN	0	Receive quality of service enable bit
		0	Receive QOS is disabled.
		1	Receive QOS is enabled.
28	RXNOCHAIN	0	Receive no buffer chaining bit
		0	Received frames can span multiple buffers.
		1	The Receive DMA controller transfers each frame into a single buffer, regardless of the frame or buffer size. All remaining frame data after the first buffer is discarded. The buffer descriptor buffer length field will contain the entire frame byte count (up to 65535 bytes).
27-25	Reserved	0	Reserved
24	RXCMFEN	0	Receive copy MAC control frames enable bit. Enables MAC control frames to be transferred to memory. MAC control frames are normally acted upon (if enabled), but not copied to memory. MAC control frames that are pause frames will be acted upon if enabled in MACCONTROL, regardless of the value of RXCMFEN. Frames transferred to memory due to RXCMFEN will have the CONTROL bit set in their EOP buffer descriptor.
		0	MAC control frames are filtered (but acted upon if enabled).
		1	MAC control frames are transferred to memory.
23	RXCSFEN	0	Receive copy short frames enable bit. Enables frames or fragments shorter than 64 bytes to be copied to memory. Frames transferred to memory due to RXCSFEN will have the FRAGMENT or UNDERSIZE bit set in their EOP buffer descriptor. Fragments are short frames that contain CRC / align / code errors and undersized are short frames without errors.
		0	Short frames are filtered.
		1	Short frames are transferred to memory.



**Table 27-60. Receive Multicast/Broadcast/Promiscuous Channel Enable Register (RXMBPENABLE)  
Field Descriptions (continued)**

Bit	Field	Value	Description
22	RXCEFEN	0 1	Receive copy error frames enable bit. Enables frames containing errors to be transferred to memory. The appropriate error bit will be set in the frame EOP buffer descriptor. Frames containing errors are filtered. Frames containing errors are transferred to memory.
21	RXCAFEN	0 1	Receive copy all frames enable bit. Enables frames that do not address match (includes multicast frames that do not hash match) to be transferred to the promiscuous channel selected by RXPROMCH bits. Such frames will be marked with the NOMATCH bit in their EOP buffer descriptor. Frames that do not address match are filtered. Frames that do not address match are transferred to the promiscuous channel selected by RXPROMCH bits.
20-19	Reserved	0	Reserved
18-16	RXPROMCH	0-7h 0 1h 2h 3h 4h 5h 6h 7h	Receive promiscuous channel select Select channel 0 to receive promiscuous frames Select channel 1 to receive promiscuous frames Select channel 2 to receive promiscuous frames Select channel 3 to receive promiscuous frames Select channel 4 to receive promiscuous frames Select channel 5 to receive promiscuous frames Select channel 6 to receive promiscuous frames Select channel 7 to receive promiscuous frames
15-14	Reserved	0	Reserved
13	RXBROADEN	0 1	Receive broadcast enable. Enable received broadcast frames to be copied to the channel selected by RXBROADCH bits. Broadcast frames are filtered. Broadcast frames are copied to the channel selected by RXBROADCH bits.
12-11	Reserved	0	Reserved
10-8	RXBROADCH	0-7h 0 1h 2h 3h 4h 5h 6h 7h	Receive broadcast channel select Select channel 0 to receive broadcast frames Select channel 1 to receive broadcast frames Select channel 2 to receive broadcast frames Select channel 3 to receive broadcast frames Select channel 4 to receive broadcast frames Select channel 5 to receive broadcast frames Select channel 6 to receive broadcast frames Select channel 7 to receive broadcast frames
7-6	Reserved	0	Reserved
5	RXMULTEN	0 1	RX multicast enable. Enable received hash matching multicast frames to be copied to the channel selected by RXMULTCH bits. Multicast frames are filtered. Multicast frames are copied to the channel selected by RXMULTCH bits.
4-3	Reserved	0	Reserved

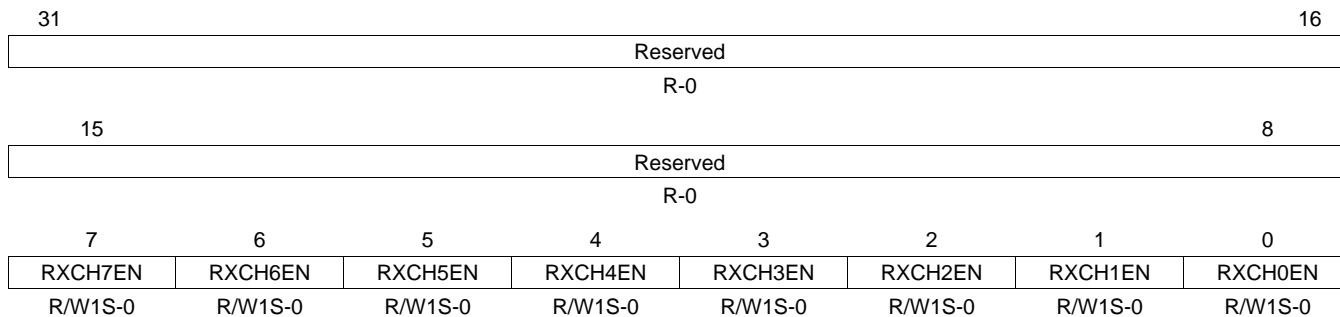
**Table 27-60. Receive Multicast/Broadcast/Promiscuous Channel Enable Register (RXMBPENABLE)  
Field Descriptions (continued)**

Bit	Field	Value	Description
2-0	RXMULTCH	0-7h	Receive multicast channel select
		0	Select channel 0 to receive multicast frames
		1h	Select channel 1 to receive multicast frames
		2h	Select channel 2 to receive multicast frames
		3h	Select channel 3 to receive multicast frames
		4h	Select channel 4 to receive multicast frames
		5h	Select channel 5 to receive multicast frames
		6h	Select channel 6 to receive multicast frames
		7h	Select channel 7 to receive multicast frames

### 27.5.22 Receive Unicast Enable Set Register (RXUNICASTSET)

The receive unicast enable set register (RXUNICASTSET) is shown in [Figure 27-60](#) and described in [Table 27-61](#).

**Figure 27-60. Receive Unicast Enable Set Register (RXUNICASTSET)**



LEGEND: R/W = Read/Write; R = Read only; W1S = Write 1 to set (writing a 0 has no effect); -n = value after reset

**Table 27-61. Receive Unicast Enable Set Register (RXUNICASTSET) Field Descriptions**

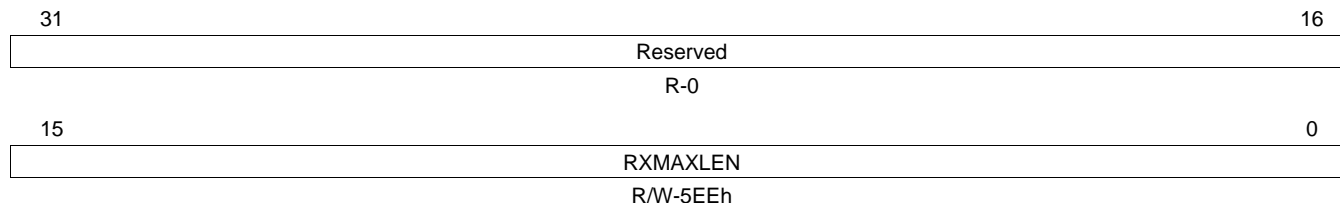
Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	RXCH7EN	0-1	Receive channel 7 unicast enable set bit. Write 1 to set the enable, a write of 0 has no effect. May be read.
6	RXCH6EN	0-1	Receive channel 6 unicast enable set bit. Write 1 to set the enable, a write of 0 has no effect. May be read.
5	RXCH5EN	0-1	Receive channel 5 unicast enable set bit. Write 1 to set the enable, a write of 0 has no effect. May be read.
4	RXCH4EN	0-1	Receive channel 4 unicast enable set bit. Write 1 to set the enable, a write of 0 has no effect. May be read.
3	RXCH3EN	0-1	Receive channel 3 unicast enable set bit. Write 1 to set the enable, a write of 0 has no effect. May be read.
2	RXCH2EN	0-1	Receive channel 2 unicast enable set bit. Write 1 to set the enable, a write of 0 has no effect. May be read.
1	RXCH1EN	0-1	Receive channel 1 unicast enable set bit. Write 1 to set the enable, a write of 0 has no effect. May be read.
0	RXCH0EN	0-1	Receive channel 0 unicast enable set bit. Write 1 to set the enable, a write of 0 has no effect. May be read.



### 27.5.24 Receive Maximum Length Register (RXMAXLEN)

The receive maximum length register (RXMAXLEN) is shown in [Figure 27-62](#) and described in [Table 27-63](#).

**Figure 27-62. Receive Maximum Length Register (RXMAXLEN)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

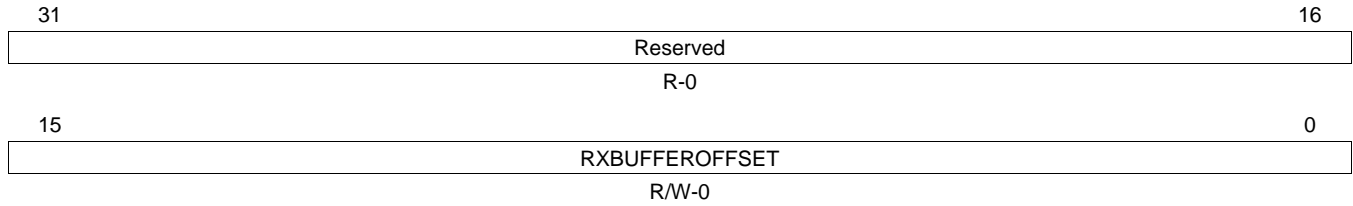
**Table 27-63. Receive Maximum Length Register (RXMAXLEN) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	RXMAXLEN	0-FFFFh	Receive maximum frame length. These bits determine the maximum length of a received frame. The reset value is 5EEh (1518). Frames with byte counts greater than RXMAXLEN are long frames. Long frames with no errors are oversized frames. Long frames with CRC, code, or alignment error are jabber frames.

**27.5.25 Receive Buffer Offset Register (RXBUFFEROFFSET)**

The receive buffer offset register (RXBUFFEROFFSET) is shown in [Figure 27-63](#) and described in [Table 27-64](#).

**Figure 27-63. Receive Buffer Offset Register (RXBUFFEROFFSET)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

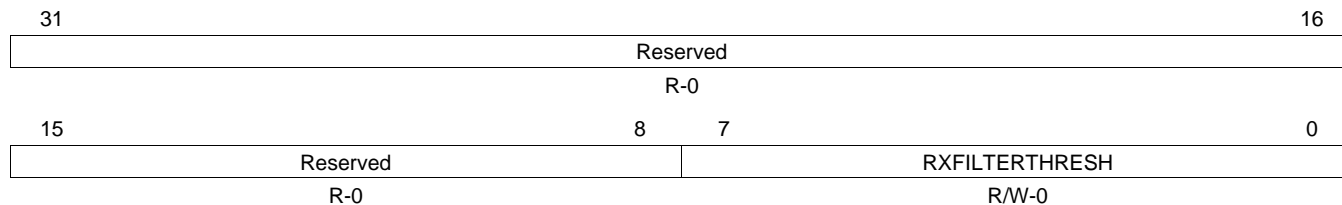
**Table 27-64. Receive Buffer Offset Register (RXBUFFEROFFSET) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	RXBUFFEROFFSET	0-FFFFh	Receive buffer offset value. These bits are written by the EMAC into each frame SOP buffer descriptor Buffer Offset field. The frame data begins after the RXBUFFEROFFSET value of bytes. A value of 0 indicates that there are no unused bytes at the beginning of the data, and that valid data begins on the first byte of the buffer. A value of Fh (15) indicates that the first 15 bytes of the buffer are to be ignored by the EMAC and that valid buffer data starts on byte 16 of the buffer. This value is used for all channels.

### 27.5.26 Receive Filter Low Priority Frame Threshold Register (RXFILTERLOWTHRESH)

The receive filter low priority frame threshold register (RXFILTERLOWTHRESH) is shown in [Figure 27-64](#) and described in [Table 27-65](#).

**Figure 27-64. Receive Filter Low Priority Frame Threshold Register (RXFILTERLOWTHRESH)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

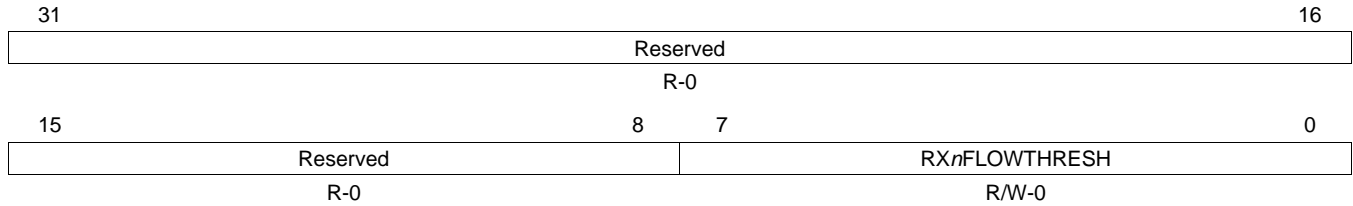
**Table 27-65. Receive Filter Low Priority Frame Threshold Register (RXFILTERLOWTHRESH) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	RXFILTERTHRESH	0-FFh	Receive filter low threshold. These bits contain the free buffer count threshold value for filtering low priority incoming frames. This field should remain 0, if no filtering is desired.

**27.5.27 Receive Channel Flow Control Threshold Registers (RX0FLOWTHRESH-RX7FLOWTHRESH)**

The receive channel 0-7 flow control threshold register (RX $n$ FLOWTHRESH) is shown in [Figure 27-65](#) and described in [Table 27-66](#).

**Figure 27-65. Receive Channel  $n$  Flow Control Threshold Register (RX $n$ FLOWTHRESH)**



LEGEND: R/W = Read/Write; R = Read only; - $n$  = value after reset

**Table 27-66. Receive Channel  $n$  Flow Control Threshold Register (RX $n$ FLOWTHRESH) Field Descriptions**

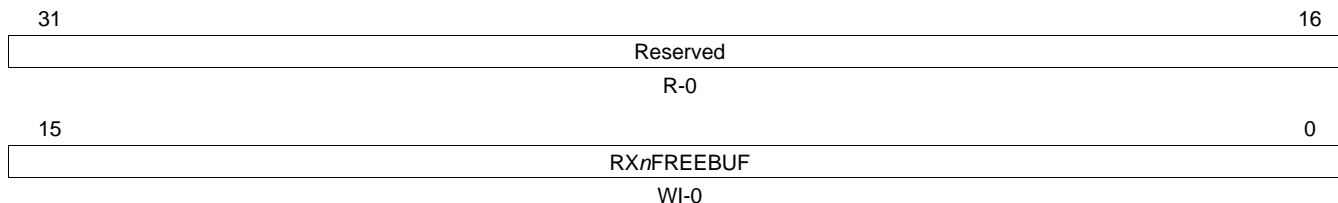
Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	RX $n$ FLOWTHRESH	0-FFh	Receive flow threshold. These bits contain the threshold value for issuing flow control on incoming frames for channel $n$ (when enabled).



### 27.5.28 Receive Channel Free Buffer Count Registers (RX0FREEBUFFER-RX7FREEBUFFER)

The receive channel 0-7 free buffer count register (RX $n$ FREEBUFFER) is shown in [Figure 27-66](#) and described in [Table 27-67](#).

**Figure 27-66. Receive Channel  $n$  Free Buffer Count Register (RX $n$ FREEBUFFER)**



LEGEND: R = Read only; WI = Write to increment; - $n$  = value after reset

**Table 27-67. Receive Channel  $n$  Free Buffer Count Register (RX $n$ FREEBUFFER) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	RX $n$ FREEBUF	0-FFh	<p>Receive free buffer count. These bits contain the count of free buffers available. The RXFILTERTHRESH value is compared with this field to determine if low priority frames should be filtered. The RX<math>n</math>FLOWTHRESH value is compared with this field to determine if receive flow control should be issued against incoming packets (if enabled). This is a write-to-increment field. This field rolls over to 0 on overflow.</p> <p>If hardware flow control or QOS is used, the host must initialize this field to the number of available buffers (one register per channel). The EMAC decrements the associated channel register for each received frame by the number of buffers in the received frame. The host must write this field with the number of buffers that have been freed due to host processing.</p>

### 27.5.29 MAC Control Register (MACCONTROL)

The MAC control register (MACCONTROL) is shown in [Figure 27-67](#) and described in [Table 27-68](#).

**Figure 27-67. MAC Control Register (MACCONTROL)**

31								16							
Reserved															
R-0															
15		14		13		12		11		10		9		8	
RMIISPEED	RXOFFLENBLOCK	RXOWNERSHIP	Rsvd	CMDIDLE	TXSHORTGAPEN	TXPTYPE	Reserved								
R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R-0								
7		6		5		4		3		2		1		0	
Reserved	TXPACE	GMIEN	TXFLOWEN	RXBUFFERFLOWEN	Reserved	LOOPBACK	FULLDUPLEX								
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0								

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 27-68. MAC Control Register (MACCONTROL) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	RMIISPEED	0	RMII interface transmit and receive speed select.
		0	Operate RMII interface in 10 Mbps speed mode.
		1	Operate RMII interface in 100 Mbps speed mode.
14	RXOFFLENBLOCK	0	Receive offset / length word write block.
		0	Do not block the DMA writes to the receive buffer descriptor offset / buffer length word.
		1	Block all EMAC DMA controller writes to the receive buffer descriptor offset / buffer length words during packet processing. When this bit is set, the EMAC will never write the third word to any receive buffer descriptor.
13	RXOWNERSHIP	0	Receive ownership write bit value.
		0	The EMAC writes the Receive ownership bit to 0 at the end of packet processing.
		1	The EMAC writes the Receive ownership bit to 1 at the end of packet processing. If you do not use the ownership mechanism, you can set this mode to preclude the necessity of software having to set this bit each time the buffer descriptor is used.
12	Reserved	0	Reserved
11	CMDIDLE	0	Command Idle bit
		0	Idle is not commanded.
		1	Idle is commanded (read IDLE in the MACSTATUS register).
10	TXSHORTGAPEN	0	Transmit Short Gap Enable
		0	Transmit with a short IPG is disabled. Normal 96-bit time IPG is inserted between packets.
		1	Transmit with a short IPG is enabled. Shorter 88-bit time IPG is inserted between packets.
9	TXPTYPE	0	Transmit queue priority type
		0	The queue uses a round-robin scheme to select the next channel for transmission.
		1	The queue uses a fixed-priority (channel 7 highest priority) scheme to select the next channel for transmission.
8-7	Reserved	0	Reserved
6	TXPACE	0	Transmit pacing enable bit
		0	Transmit pacing is disabled.
		1	Transmit pacing is enabled.
5	GMIEN	0	GMII enable bit
		0	GMII RX and TX are held in reset.
		1	GMII RX and TX are enabled for receive and transmit.

**Table 27-68. MAC Control Register (MACCONTROL) Field Descriptions (continued)**

Bit	Field	Value	Description
4	TXFLOWEN		Transmit flow control enable bit. This bit determines if incoming pause frames are acted upon in full-duplex mode. Incoming pause frames are not acted upon in half-duplex mode, regardless of this bit setting. The RXMBPENABLE bits determine whether or not received pause frames are transferred to memory.
		0	Transmit flow control is disabled. Full-duplex mode: incoming pause frames are not acted upon.
		1	Transmit flow control is enabled. Full-duplex mode: incoming pause frames are acted upon.
3	RXBUFFERFLOWEN	0	Receive buffer flow control enable bit Receive flow control is disabled. Half-duplex mode: no flow control generated collisions are sent. Full-duplex mode: no outgoing pause frames are sent.
		1	Receive flow control is enabled. Half-duplex mode: collisions are initiated when receive buffer flow control is triggered. Full-duplex mode: outgoing pause frames are sent when receive flow control is triggered.
2	Reserved	0	Reserved
1	LOOPBACK		Loopback mode. The loopback mode forces internal full-duplex mode regardless of the FULLDUPLEX bit. The loopback bit should be changed only when GMIEN bit is deasserted.
		0	Loopback mode is disabled.
		1	Loopback mode is enabled.
0	FULLDUPLEX		Full duplex mode.
		0	Half-duplex mode is enabled.
		1	Full-duplex mode is enabled.

### 27.5.30 MAC Status Register (MACSTATUS)

The MAC status register (MACSTATUS) is shown in [Figure 27-68](#) and described in [Table 27-69](#).

**Figure 27-68. MAC Status Register (MACSTATUS)**

31	30	24	23	20	19	18	16
IDLE	Reserved		TXERRCODE		Rsvd	TXERRCH	
R-0	R-0		R-0		R-0	R-0	
15	12	11	10	8			
RXERRCODE			Reserved	RXERRCH			
R-0			R-0	R-0			
7	3		2	1	0		
Reserved				RXQOSACT	RXFLOWACT	TXFLOWACT	
R-0				R-0	R-0	R-0	

LEGEND: R = Read only; -n = value after reset

**Table 27-69. MAC Status Register (MACSTATUS) Field Descriptions**

Bit	Field	Value	Description
31	IDLE	0 1	EMAC idle bit. This bit is cleared to 0 at reset; one clock after reset, it goes to 1. The EMAC is not idle. The EMAC is in the idle state.
30-24	Reserved	0	Reserved
23-20	TXERRCODE	0-Fh 0 1h 2h 3h 4h 5h 6h 7h-Fh	Transmit host error code. These bits indicate that EMAC detected transmit DMA related host errors. The host should read this field after a host error interrupt (HOSTPEND) to determine the error. Host error interrupts require hardware reset in order to recover. A 0 packet length is an error, but it is not detected. No error SOP error; the buffer is the first buffer in a packet, but the SOP bit is not set in software. Ownership bit not set in SOP buffer Zero next buffer descriptor pointer without EOP Zero buffer pointer Zero buffer length Packet length error (sum of buffers is less than packet length) Reserved
19	Reserved	0	Reserved
18-16	TXERRCH	0-7h 0 1h 2h 3h 4h 5h 6h 7h	Transmit host error channel. These bits indicate which transmit channel the host error occurred on. This field is cleared to 0 on a host read. The host error occurred on transmit channel 0 The host error occurred on transmit channel 1 The host error occurred on transmit channel 2 The host error occurred on transmit channel 3 The host error occurred on transmit channel 4 The host error occurred on transmit channel 5 The host error occurred on transmit channel 6 The host error occurred on transmit channel 7

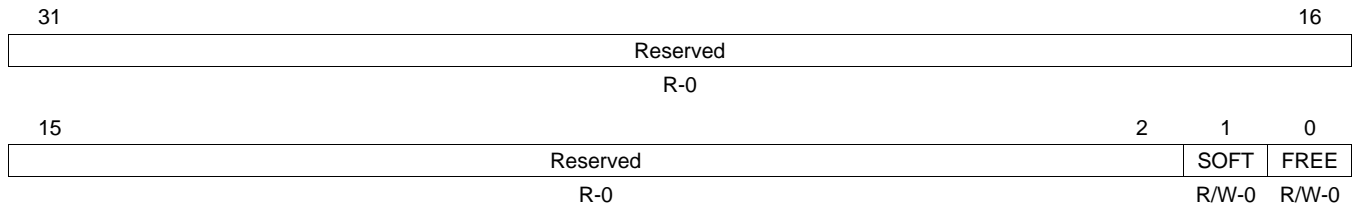
**Table 27-69. MAC Status Register (MACSTATUS) Field Descriptions (continued)**

Bit	Field	Value	Description
15-12	RXERRCODE	0-Fh 0 1h 2h 3h 4h 5h-Fh	Receive host error code. These bits indicate that EMAC detected receive DMA related host errors. The host should read this field after a host error interrupt (HOSTPEND) to determine the error. Host error interrupts require hardware reset in order to recover. No error Reserved Ownership bit not set in SOP buffer Reserved Zero buffer pointer Reserved
11	Reserved	0	Reserved
10-8	RXERRCH	0-7h 0 1h 2h 3h 4h 5h 6h 7h	Receive host error channel. These bits indicate which receive channel the host error occurred on. This field is cleared to 0 on a host read. The host error occurred on receive channel 0 The host error occurred on receive channel 1 The host error occurred on receive channel 2 The host error occurred on receive channel 3 The host error occurred on receive channel 4 The host error occurred on receive channel 5 The host error occurred on receive channel 6 The host error occurred on receive channel 7
7-3	Reserved	0	Reserved
2	RXQOSACT	0 1	Receive Quality of Service (QOS) active bit. When asserted, indicates that receive quality of service is enabled and that at least one channel freebuffer count (RX <sub>n</sub> FREEBUFFER) is less than or equal to the RXFILTERLOWTHRESH value. Receive quality of service is disabled. Receive quality of service is enabled.
1	RXFLOWACT	0 1	Receive flow control active bit. When asserted, at least one channel freebuffer count (RX <sub>n</sub> FREEBUFFER) is less than or equal to the channel's corresponding RX <sub>n</sub> FILTERTHRESH value. Receive flow control is inactive. Receive flow control is active.
0	TXFLOWACT	0 1	Transmit flow control active bit. When asserted, this bit indicates that the pause time period is being observed for a received pause frame. No new transmissions will begin while this bit is asserted, except for the transmission of pause frames. Any transmission in progress when this bit is asserted will complete. Transmit flow control is inactive. Transmit flow control is active.

### 27.5.31 Emulation Control Register (EMCONTROL)

The emulation control register (EMCONTROL) is shown in [Figure 27-69](#) and described in [Table 27-70](#).

**Figure 27-69. Emulation Control Register (EMCONTROL)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

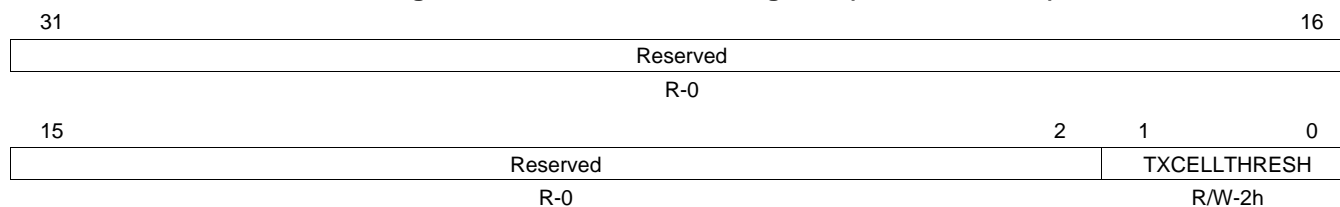
**Table 27-70. Emulation Control Register (EMCONTROL) Field Descriptions**

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	SOFT	0	Emulation soft bit. This bit is used in conjunction with FREE bit to determine the emulation suspend mode. This bit has no effect if FREE = 1. Soft mode is disabled. EMAC stops immediately during emulation halt.
		1	Soft mode is enabled. During emulation halt, EMAC stops after completion of current operation.
0	FREE	0	Emulation free bit. This bit is used in conjunction with SOFT bit to determine the emulation suspend mode. Free-running mode is disabled. During emulation halt, SOFT bit determines operation of EMAC.
		1	Free-running mode is enabled. During emulation halt, EMAC continues to operate.

### 27.5.32 FIFO Control Register (FIFOCONTROL)

The FIFO control register (FIFOCONTROL) is shown in [Figure 27-70](#) and described in [Table 27-71](#).

**Figure 27-70. FIFO Control Register (FIFOCONTROL)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 27-71. FIFO Control Register (FIFOCONTROL) Field Descriptions**

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1-0	TXCELLTHRESH	0-3h	Transmit FIFO cell threshold. Indicates the number of 64-byte packet cells required to be in the transmit FIFO before the packet transfer is initiated. Packets with fewer cells will be initiated when the complete packet is contained in the FIFO. The default value is 2, but 3 is also valid. 0 and 1 are not valid values.
		0-1h	Not a valid value.
		2h	Two 64-byte packet cells required to be in the transmit FIFO.
		3h	Three 64-byte packet cells required to be in the transmit FIFO.

### 27.5.33 MAC Configuration Register (MACCONFIG)

The MAC configuration register (MACCONFIG) is shown in [Figure 27-71](#) and described in [Table 27-72](#).

**Figure 27-71. MAC Configuration Register (MACCONFIG)**

31	24	23	16
TXCELLDEPTH		RXCELLDEPTH	
R-3h		R-3h	
15	8	7	0
ADDRESSTYPE		MACCFIG	
R-2h		R-2h	

LEGEND: R = Read only; -n = value after reset

**Table 27-72. MAC Configuration Register (MACCONFIG) Field Descriptions**

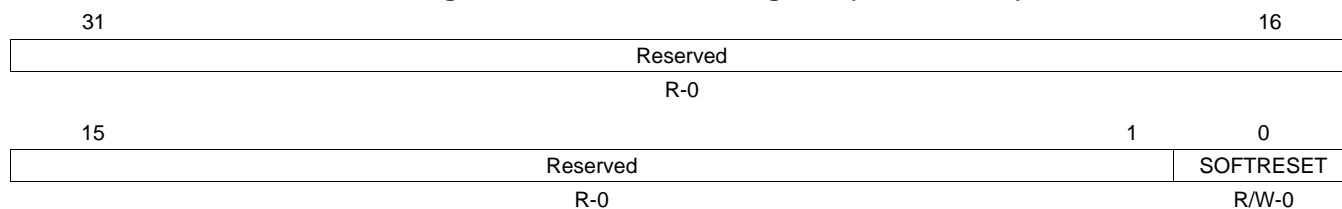
Bit	Field	Value	Description
31-24	TXCELLDEPTH	3h	Transmit cell depth. These bits indicate the number of cells in the transmit FIFO.
23-16	RXCELLDEPTH	3h	Receive cell depth. These bits indicate the number of cells in the receive FIFO.
15-8	ADDRESSTYPE	2h	Address type
7-0	MACCFIG	2h	MAC configuration value



### 27.5.34 Soft Reset Register (SOFTRESET)

The soft reset register (SOFTRESET) is shown in [Figure 27-72](#) and described in [Table 27-73](#).

**Figure 27-72. Soft Reset Register (SOFTRESET)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

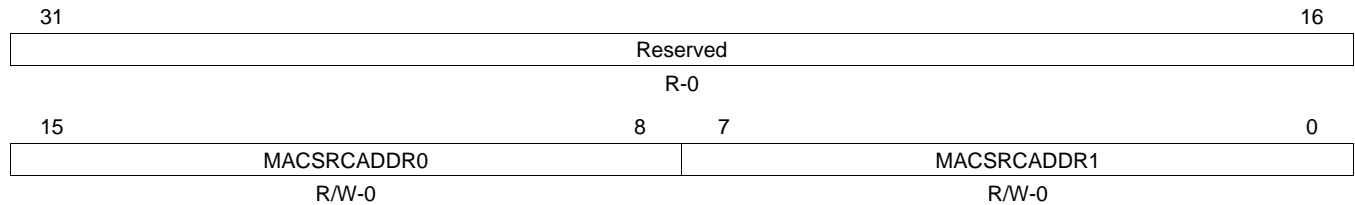
**Table 27-73. Soft Reset Register (SOFTRESET) Field Descriptions**

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	SOFTRESET	0	Software reset. Writing a 1 to this bit causes the EMAC logic to be reset. Software reset occurs when the receive and transmit DMA controllers are in an idle state to avoid locking up the Configuration bus. After writing a 1 to this bit, it may be polled to determine if the reset has occurred. If a 1 is read, the reset has not yet occurred. If a 0 is read, then a reset has occurred.
		0	A software reset has not occurred.
		1	A software reset has occurred.

### 27.5.35 MAC Source Address Low Bytes Register (MACSRCADDRLO)

The MAC source address low bytes register (MACSRCADDRLO) is shown in [Figure 27-73](#) and described in [Table 27-74](#).

**Figure 27-73. MAC Source Address Low Bytes Register (MACSRCADDRLO)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

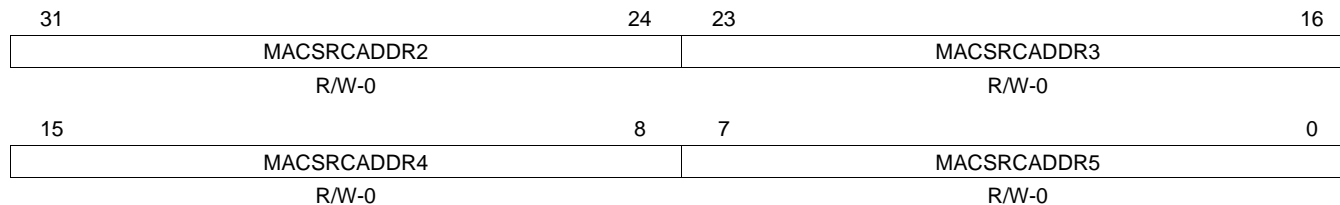
**Table 27-74. MAC Source Address Low Bytes Register (MACSRCADDRLO) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-8	MACSRCADDR0	0-FFh	MAC source address lower 8-0 bits (byte 0)
7-0	MACSRCADDR1	0-FFh	MAC source address bits 15-8 (byte 1)

### 27.5.36 MAC Source Address High Bytes Register (MACSRCADDRHI)

The MAC source address high bytes register (MACSRCADDRHI) is shown in [Figure 27-74](#) and described in [Table 27-75](#).

**Figure 27-74. MAC Source Address High Bytes Register (MACSRCADDRHI)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 27-75. MAC Source Address High Bytes Register (MACSRCADDRHI) Field Descriptions**

Bit	Field	Value	Description
31-24	MACSRCADDR2	0-FFh	MAC source address bits 23-16 (byte 2)
23-16	MACSRCADDR3	0-FFh	MAC source address bits 31-24 (byte 3)
15-8	MACSRCADDR4	0-FFh	MAC source address bits 39-32 (byte 4)
7-0	MACSRCADDR5	0-FFh	MAC source address bits 47-40 (byte 5)

**27.5.37 MAC Hash Address Register 1 (MACHASH1)**

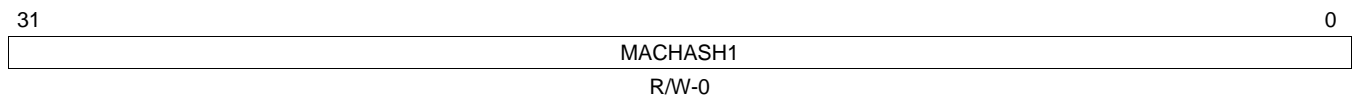
The MAC hash registers allow group addressed frames to be accepted on the basis of a hash function of the address. The hash function creates a 6-bit data value (Hash\_fun) from the 48-bit destination address (DA) as follows:

```
Hash_fun(0)=DA(0) XOR DA(6) XOR DA(12) XOR DA(18) XOR DA(24) XOR DA(30) XOR DA(36) XOR DA(42);
Hash_fun(1)=DA(1) XOR DA(7) XOR DA(13) XOR DA(19) XOR DA(25) XOR DA(31) XOR DA(37) XOR DA(43);
Hash_fun(2)=DA(2) XOR DA(8) XOR DA(14) XOR DA(20) XOR DA(26) XOR DA(32) XOR DA(38) XOR DA(44);
Hash_fun(3)=DA(3) XOR DA(9) XOR DA(15) XOR DA(21) XOR DA(27) XOR DA(33) XOR DA(39) XOR DA(45);
Hash_fun(4)=DA(4) XOR DA(10) XOR DA(16) XOR DA(22) XOR DA(28) XOR DA(34) XOR DA(40) XOR DA(46);
Hash_fun(5)=DA(5) XOR DA(11) XOR DA(17) XOR DA(23) XOR DA(29) XOR DA(35) XOR DA(41) XOR DA(47);
```

This function is used as an offset into a 64-bit hash table stored in MACHASH1 and MACHASH2 that indicates whether a particular address should be accepted or not.

The MAC hash address register 1 (MACHASH1) is shown in [Figure 27-75](#) and described in [Table 27-76](#).

**Figure 27-75. MAC Hash Address Register 1 (MACHASH1)**



LEGEND: R/W = Read/Write; -n = value after reset

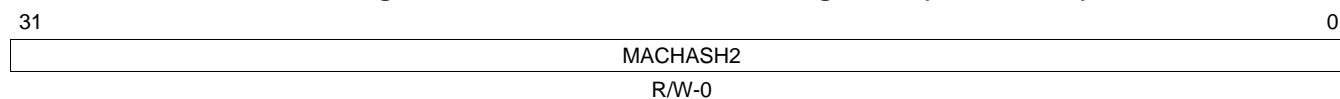
**Table 27-76. MAC Hash Address Register 1 (MACHASH1) Field Descriptions**

Bit	Field	Value	Description
31-0	MACHASH1	0-FFFF FFFFh	Least-significant 32 bits of the hash table corresponding to hash values 0 to 31. If a hash table bit is set, then a group address that hashes to that bit index is accepted.

### 27.5.38 MAC Hash Address Register 2 (MACHASH2)

The MAC hash address register 2 (MACHASH2) is shown in [Figure 27-76](#) and described in [Table 27-77](#).

**Figure 27-76. MAC Hash Address Register 2 (MACHASH2)**



LEGEND: R/W = Read/Write; -n = value after reset

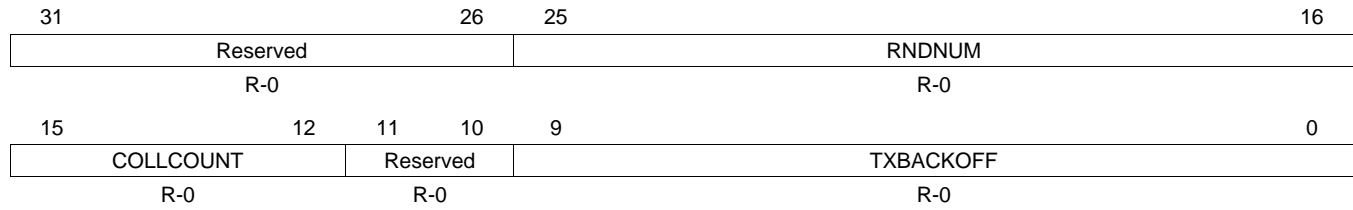
**Table 27-77. MAC Hash Address Register 2 (MACHASH2) Field Descriptions**

Bit	Field	Value	Description
31-0	MACHASH2	0-FFFF FFFFh	Most-significant 32 bits of the hash table corresponding to hash values 32 to 63. If a hash table bit is set, then a group address that hashes to that bit index is accepted.

### 27.5.39 Back Off Test Register (BOFFTEST)

The back off test register (BOFFTEST) is shown in [Figure 27-77](#) and described in [Table 27-78](#).

**Figure 27-77. Back Off Random Number Generator Test Register (BOFFTEST)**



LEGEND: R = Read only; -n = value after reset

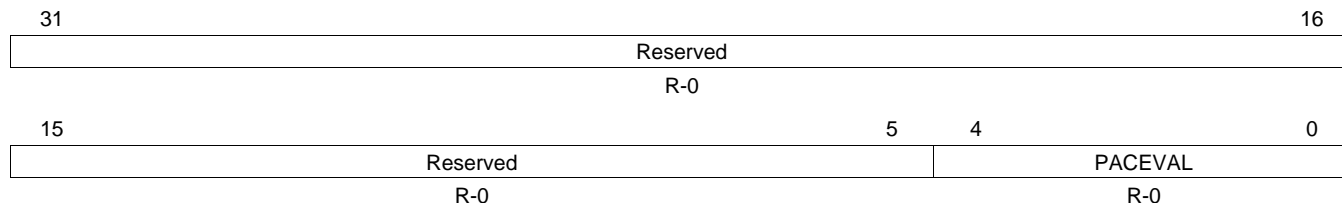
**Table 27-78. Back Off Test Register (BOFFTEST) Field Descriptions**

Bit	Field	Value	Description
31-26	Reserved	0	Reserved
25-16	RNDNUM	0-3FFh	Backoff random number generator. This field allows the Backoff Random Number Generator to be read. Reading this field returns the generator's current value. The value is reset to 0 and begins counting on the clock after the deassertion of reset.
15-12	COLLCOUNT	0-Fh	Collision count. These bits indicate the number of collisions the current frame has experienced.
11-10	Reserved	0	Reserved
9-0	TXBACKOFF	0-3FFh	Backoff count. This field allows the current value of the backoff counter to be observed for test purposes. This field is loaded automatically according to the backoff algorithm, and is decremented by one for each slot time after the collision.

### 27.5.40 Transmit Pacing Algorithm Test Register (TPACETEST)

The transmit pacing algorithm test register (TPACETEST) is shown in [Figure 27-78](#) and described in [Table 27-79](#).

**Figure 27-78. Transmit Pacing Algorithm Test Register (TPACETEST)**



LEGEND: R = Read only; -n = value after reset

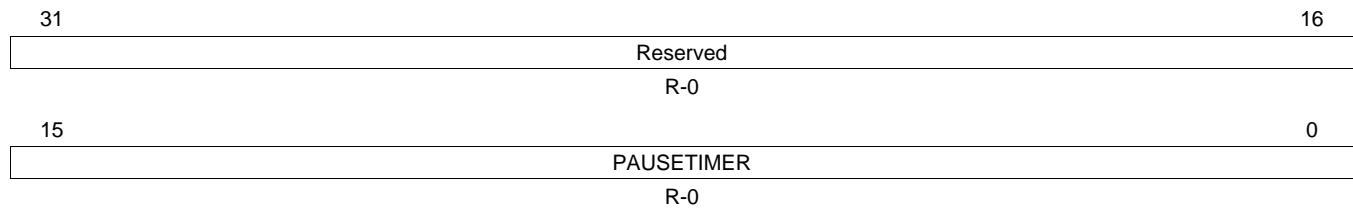
**Table 27-79. Transmit Pacing Algorithm Test Register (TPACETEST) Field Descriptions**

Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4-0	PACEVAL	0-1Fh	Pacing register current value. A nonzero value in this field indicates that transmit pacing is active. A transmit frame collision or deferral causes PACEVAL to be loaded with 1Fh (31); good frame transmissions (with no collisions or deferrals) cause PACEVAL to be decremented down to 0. When PACEVAL is nonzero, the transmitter delays four Inter Packet Gaps between new frame transmissions after each successfully transmitted frame that had no deferrals or collisions. If a transmit frame is deferred or suffers a collision, the IPG time is not stretched to four times the normal value. Transmit pacing helps reduce capture effects, which improves overall network bandwidth.

### 27.5.41 Receive Pause Timer Register (RXPAUSE)

The receive pause timer register (RXPAUSE) is shown in [Figure 27-79](#) and described in [Table 27-80](#).

**Figure 27-79. Receive Pause Timer Register (RXPAUSE)**



LEGEND: R = Read only; -n = value after reset

**Table 27-80. Receive Pause Timer Register (RXPAUSE) Field Descriptions**

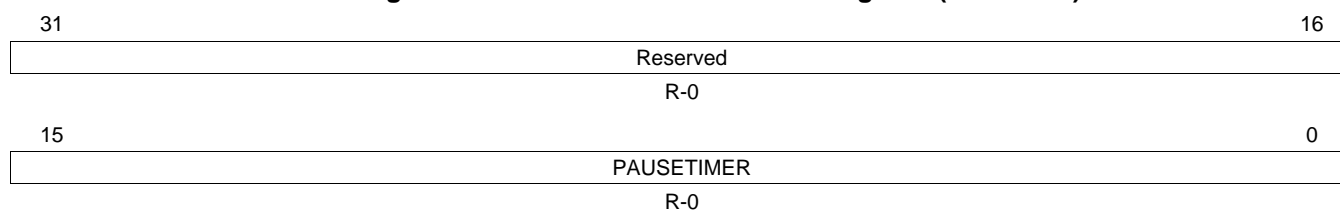
Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	PAUSETIMER	0-FFh	Receive pause timer value. These bits allow the contents of the receive pause timer to be observed. The receive pause timer is loaded with FF00h when the EMAC sends an outgoing pause frame (with pause time of FFFFh). The receive pause timer is decremented at slot time intervals. If the receive pause timer decrements to 0, then another outgoing pause frame is sent and the load/decrement process is repeated.



### 27.5.42 Transmit Pause Timer Register (TXPAUSE)

The transmit pause timer register (TXPAUSE) is shown in [Figure 27-80](#) and described in [Table 27-81](#).

**Figure 27-80. Transmit Pause Timer Register (TXPAUSE)**



LEGEND: R = Read only; -n = value after reset

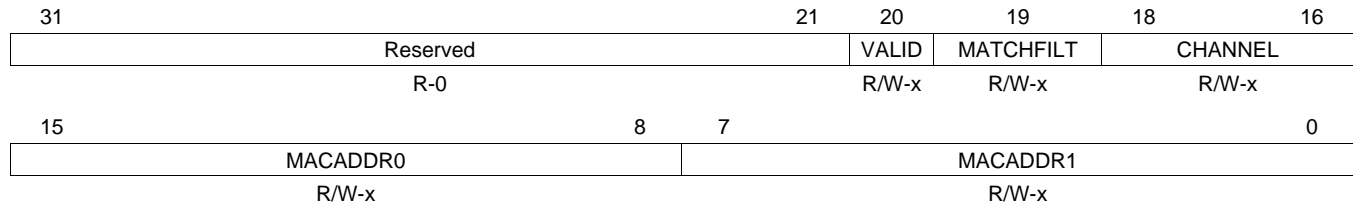
**Table 27-81. Transmit Pause Timer Register (TXPAUSE) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	PAUSETIMER	0-FFh	Transmit pause timer value. These bits allow the contents of the transmit pause timer to be observed. The transmit pause timer is loaded by a received (incoming) pause frame, and then decremented at slot time intervals down to 0, at which time EMAC transmit frames are again enabled.

### 27.5.43 MAC Address Low Bytes Register (MACADDRLO)

The MAC address low bytes register used in address matching (MACADDRLO), is shown in [Figure 27-81](#) and described in [Table 27-82](#).

**Figure 27-81. MAC Address Low Bytes Register (MACADDRLO)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset; -x = value is indeterminate after reset

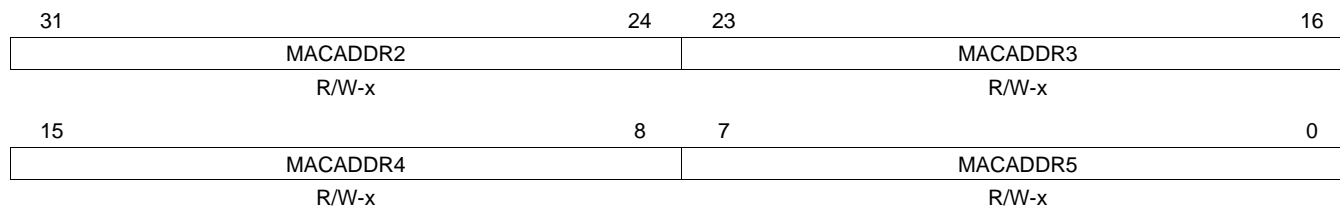
**Table 27-82. MAC Address Low Bytes Register (MACADDRLO) Field Descriptions**

Bit	Field	Value	Description
31-21	Reserved	0	Reserved
20	VALID	0 1	Address valid bit. This bit should be cleared to zero for unused address channels Address is not valid and will not be used for matching or filtering incoming packets Address is valid and will be used for matching or filtering incoming packets
19	MATCHFILT	0 1	Match or filter bit The address will be used (if the VALID bit is set) to filter incoming packet addresses The address will be used (if the VALID bit is set) to match incoming packet addresses
18-16	CHANNEL	0-7h	Channel select. Determines which receive channel a valid address match will be transferred to. The channel is a don't care if MATCHFILT is cleared to 0.
15-8	MACADDR0	0-FFh	MAC address lower 8-0 bits (byte 0)
7-0	MACADDR1	0-FFh	MAC address bits 15-8 (byte 1)

### 27.5.44 MAC Address High Bytes Register (MACADDRHI)

The MAC address high bytes register (MACADDRHI) is shown in [Figure 27-82](#) and described in [Table 27-83](#).

**Figure 27-82. MAC Address High Bytes Register (MACADDRHI)**



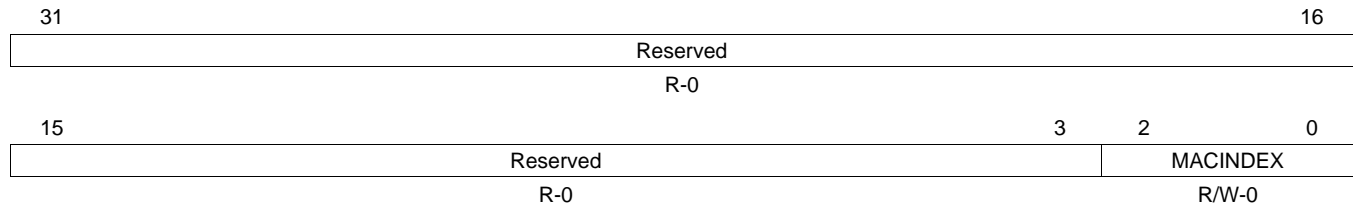
LEGEND: R/W = Read/Write; -x = value is indeterminate after reset

**Table 27-83. MAC Address High Bytes Register (MACADDRHI) Field Descriptions**

Bit	Field	Value	Description
31-24	MACADDR2	0-FFh	MAC source address bits 23-16 (byte 2)
23-16	MACADDR3	0-FFh	MAC source address bits 31-24 (byte 3)
15-8	MACADDR4	0-FFh	MAC source address bits 39-32 (byte 4)
7-0	MACADDR5	0-FFh	MAC source address bits 47-40 (byte 5). Bit 40 is the group bit. It is forced to 0 and read as 0. Therefore, only unicast addresses are represented in the address table.

**27.5.45 MAC Index Register (MACINDEX)**

The MAC index register (MACINDEX) is shown in [Figure 27-83](#) and described in [Table 27-84](#).

**Figure 27-83. MAC Index Register (MACINDEX)**


LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

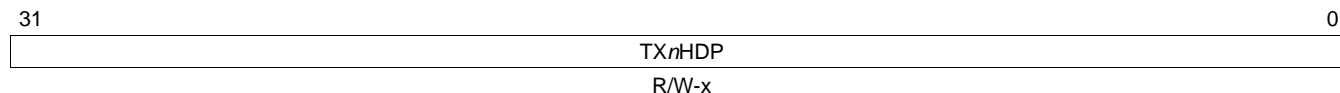
**Table 27-84. MAC Index Register (MACINDEX) Field Descriptions**

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2-0	MACINDEX	0-7h	MAC address index. All eight addresses share the upper 40 bits. Only the lower byte is unique for each address. An address is written by first writing the address number (channel) into the MACINDEX register. The upper 32 bits of the address are then written to the MACADDRHI register, which is followed by writing the lower 16 bits of the address to the MACADDRLO register. Since all eight addresses share the upper 40 bits of the address, the MACADDRHI register only needs to be written the first time.

### 27.5.46 Transmit Channel DMA Head Descriptor Pointer Registers (TX0HDP-TX7HDP)

The transmit channel 0-7 DMA head descriptor pointer register (TX $n$ HDP) is shown in [Figure 27-84](#) and described in [Table 27-85](#).

**Figure 27-84. Transmit Channel  $n$  DMA Head Descriptor Pointer Register (TX $n$ HDP)**



LEGEND: R/W = Read/Write; - $n$  = value after reset; -x = value is indeterminate after reset

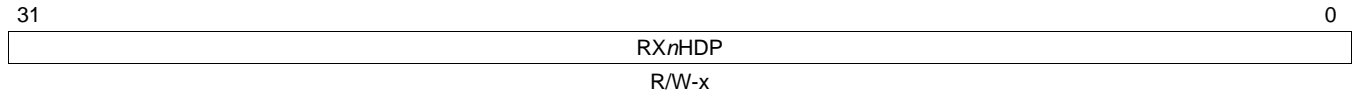
**Table 27-85. Transmit Channel  $n$  DMA Head Descriptor Pointer Register (TX $n$ HDP)  
Field Descriptions**

Bit	Field	Value	Description
31-0	TX $n$ HDP	0-FFFF FFFFh	Transmit channel $n$ DMA Head Descriptor pointer. Writing a transmit DMA buffer descriptor address to a head pointer location initiates transmit DMA operations in the queue for the selected channel. Writing to these locations when they are nonzero is an error (except at reset). Host software must initialize these locations to 0 on reset.

**27.5.47 Receive Channel DMA Head Descriptor Pointer Registers (RX0HDP-RX7HDP)**

The receive channel 0-7 DMA head descriptor pointer register (RXnHDP) is shown in [Figure 27-85](#) and described in [Table 27-86](#).

**Figure 27-85. Receive Channel *n* DMA Head Descriptor Pointer Register (RXnHDP)**



LEGEND: R/W = Read/Write; -n = value after reset; -x = value is indeterminate after reset

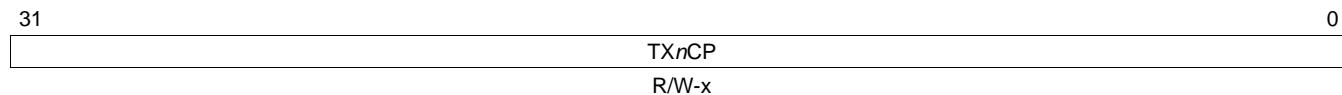
**Table 27-86. Receive Channel *n* DMA Head Descriptor Pointer Register (RXnHDP) Field Descriptions**

Bit	Field	Value	Description
31-0	RXnHDP	0-FFFF FFFFh	Receive channel <i>n</i> DMA Head Descriptor pointer. Writing a receive DMA buffer descriptor address to this location allows receive DMA operations in the selected channel when a channel frame is received. Writing to these locations when they are nonzero is an error (except at reset). Host software must initialize these locations to 0 on reset.

### 27.5.48 Transmit Channel Completion Pointer Registers (TX0CP-TX7CP)

The transmit channel 0-7 completion pointer register (TX $n$ CP) is shown in [Figure 27-86](#) and described in [Table 27-87](#).

**Figure 27-86. Transmit Channel  $n$  Completion Pointer Register (TX $n$ CP)**



LEGEND: R/W = Read/Write; - $n$  = value after reset; -x = value is indeterminate after reset

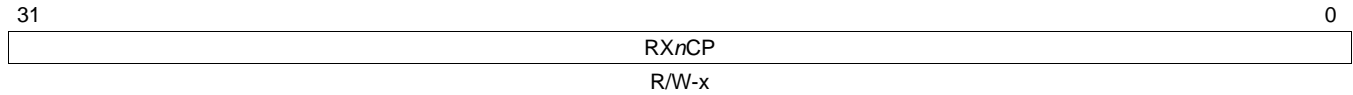
**Table 27-87. Transmit Channel  $n$  Completion Pointer Register (TX $n$ CP) Field Descriptions**

Bit	Field	Value	Description
31-0	TX $n$ CP	0-FFFF FFFFh	Transmit channel $n$ completion pointer register is written by the host with the buffer descriptor address for the last buffer processed by the host during interrupt processing. The EMAC uses the value written to determine if the interrupt should be deasserted.

**27.5.49 Receive Channel Completion Pointer Registers (RX0CP-RX7CP)**

The receive channel 0-7 completion pointer register (RX $n$ CP) is shown in [Figure 27-87](#) and described in [Table 27-88](#).

**Figure 27-87. Receive Channel  $n$  Completion Pointer Register (RX $n$ CP)**



LEGEND: R/W = Read/Write; - $n$  = value after reset; -x = value is indeterminate after reset

**Table 27-88. Receive Channel  $n$  Completion Pointer Register (RX $n$ CP) Field Descriptions**

Bit	Field	Value	Description
31-0	RX $n$ CP	0-FFFF FFFFh	Receive channel $n$ completion pointer register is written by the host with the buffer descriptor address for the last buffer processed by the host during interrupt processing. The EMAC uses the value written to determine if the interrupt should be deasserted.

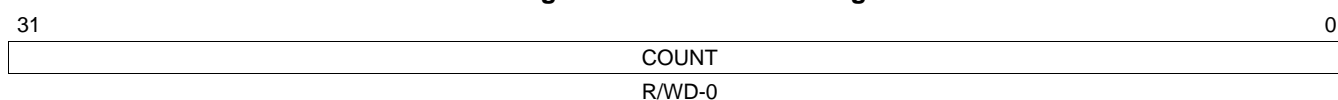


## 27.5.50 Network Statistics Registers

The EMAC has a set of statistics that record events associated with frame traffic. The statistics values are cleared to zero 38 clocks after the rising edge of reset. When the GMIIEN bit in the MACCONTROL register is set, all statistics registers (see [Figure 27-88](#)) are write-to-decrement. The value written is subtracted from the register value with the result stored in the register. If a value greater than the statistics value is written, then zero is written to the register (writing FFFF FFFFh clears a statistics location). When the GMIIEN bit is cleared, all statistics registers are read/write (normal write direct, so writing 0000 0000h clears a statistics location). All write accesses must be 32-bit accesses.

The statistics interrupt (STATPEND) is issued, if enabled, when any statistics value is greater than or equal to 8000 0000h. The statistics interrupt is removed by writing to decrement any statistics value greater than 8000 0000h. The statistics are mapped into internal memory space and are 32-bits wide. All statistics rollover from FFFF FFFFh to 0000 0000h.

**Figure 27-88. Statistics Register**



LEGEND: R/W = Read/Write; WD = Write to decrement; -n = value after reset

### 27.5.50.1 Good Receive Frames Register (RXGOODFRAMES)

The total number of good frames received on the EMAC. A good frame is defined as having all of the following:

- Any data or MAC control frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- Was of length 64 to RXMAXLEN bytes inclusive
- Had no CRC error, alignment error, or code error

See [Section 27.2.6.5](#) for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

### 27.5.50.2 Broadcast Receive Frames Register (RXBCASTFRAMES)

The total number of good broadcast frames received on the EMAC. A good broadcast frame is defined as having all of the following:

- Any data or MAC control frame that was destined for address FF-FF-FF-FF-FF-FFh only
- Was of length 64 to RXMAXLEN bytes inclusive
- Had no CRC error, alignment error, or code error

See [Section 27.2.6.5](#) for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

### 27.5.50.3 Multicast Receive Frames Register (RXMCASTFRAMES)

The total number of good multicast frames received on the EMAC. A good multicast frame is defined as having all of the following:

- Any data or MAC control frame that was destined for any multicast address other than FF-FF-FF-FF-FF-FFh
- Was of length 64 to RXMAXLEN bytes inclusive
- Had no CRC error, alignment error, or code error

See [Section 27.2.6.5](#) for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

#### 27.5.50.4 Pause Receive Frames Register (RXPAUSEFRAMES)

The total number of IEEE 802.3X pause frames received by the EMAC (whether acted upon or not). A pause frame is defined as having all of the following:

- Contained any unicast, broadcast, or multicast address
- Contained the length/type field value 88.08h and the opcode 0001h
- Was of length 64 to RXMAXLEN bytes inclusive
- Had no CRC error, alignment error, or code error
- Pause-frames had been enabled on the EMAC (TXFLOWEN bit is set in MACCONTROL).

The EMAC could have been in either half-duplex or full-duplex mode. See [Section 27.2.6.5](#) for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

#### 27.5.50.5 Receive CRC Errors Register (RXCRCERRORS)

The total number of frames received on the EMAC that experienced a CRC error. A frame with CRC errors is defined as having all of the following:

- Was any data or MAC control frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- Was of length 64 to RXMAXLEN bytes inclusive
- Had no alignment or code error
- Had a CRC error. A CRC error is defined as having all of the following:
  - A frame containing an even number of nibbles
  - Fails the frame check sequence test

See [Section 27.2.6.5](#) for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

#### 27.5.50.6 Receive Alignment/Code Errors Register (RXALIGNCODEERRORS)

The total number of frames received on the EMAC that experienced an alignment error or code error. Such a frame is defined as having all of the following:

- Was any data or MAC control frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- Was of length 64 to RXMAXLEN bytes inclusive
- Had either an alignment error or a code error
  - An alignment error is defined as having all of the following:
    - A frame containing an odd number of nibbles
    - Fails the frame check sequence test, if the final nibble is ignored
  - A code error is defined as a frame that has been discarded because the EMACs MII\_RXER pin is driven with a one for at least one bit-time's duration at any point during the frame's reception.

Overruns have no effect on this statistic.

CRC alignment or code errors can be calculated by summing receive alignment errors, receive code errors, and receive CRC errors.

### 27.5.50.7 Receive Oversized Frames Register (RXOVERSIZED)

The total number of oversized frames received on the EMAC. An oversized frame is defined as having all of the following:

- Was any data or MAC control frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- Was greater than RXMAXLEN in bytes
- Had no CRC error, alignment error, or code error

See [Section 27.2.6.5](#) for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

### 27.5.50.8 Receive Jabber Frames Register (RXJABBER)

The total number of jabber frames received on the EMAC. A jabber frame is defined as having all of the following:

- Was any data or MAC control frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- Was greater than RXMAXLEN bytes long
- Had a CRC error, alignment error, or code error

See [Section 27.2.6.5](#) for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

### 27.5.50.9 Receive Undersized Frames Register (RXUNDERSIZED)

The total number of undersized frames received on the EMAC. An undersized frame is defined as having all of the following:

- Was any data frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- Was less than 64 bytes long
- Had no CRC error, alignment error, or code error

See [Section 27.2.6.5](#) for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

### 27.5.50.10 Receive Frame Fragments Register (RXFRAGMENTS)

The total number of frame fragments received on the EMAC. A frame fragment is defined as having all of the following:

- Any data frame (address matching does not matter)
- Was less than 64 bytes long
- Had a CRC error, alignment error, or code error
- Was not the result of a collision caused by half duplex, collision based flow control

See [Section 27.2.6.5](#) for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

### 27.5.50.11 Filtered Receive Frames Register (RXFILTERED)

The total number of frames received on the EMAC that the EMAC address matching process indicated should be discarded. Such a frame is defined as having all of the following:

- Was any data frame (not MAC control frame) destined for any unicast, broadcast, or multicast address
- Did not experience any CRC error, alignment error, code error
- The address matching process decided that the frame should be discarded (filtered) because it did not match the unicast, broadcast, or multicast address, and it did not match due to promiscuous mode.

To determine the number of receive frames discarded by the EMAC for any reason, sum the following statistics (promiscuous mode disabled):

- Receive fragments
- Receive undersized frames
- Receive CRC errors
- Receive alignment/code errors
- Receive jabbers
- Receive overruns
- Receive filtered frames

This may not be an exact count because the receive overruns statistic is independent of the other statistics, so if an overrun occurs at the same time as one of the other discard reasons, then the above sum double-counts that frame.

#### **27.5.50.12 Receive QOS Filtered Frames Register (RXQOSFILTERED)**

The total number of frames received on the EMAC that were filtered due to receive quality of service (QOS) filtering. Such a frame is defined as having all of the following:

- Any data or MAC control frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- The frame destination channel flow control threshold register (RX $n$ FLOWTHRESH) value was greater than or equal to the channel's corresponding free buffer register (RX $n$ FREEBUFFER) value
- Was of length 64 to RXMAXLEN
- RXQOSEN bit is set in RXMBPENABLE
- Had no CRC error, alignment error, or code error

See [Section 27.2.6.5](#) for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

#### **27.5.50.13 Receive Octet Frames Register (RXOCTETS)**

The total number of bytes in all good frames received on the EMAC. A good frame is defined as having all of the following:

- Any data or MAC control frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- Was of length 64 to RXMAXLEN bytes inclusive
- Had no CRC error, alignment error, or code error

See [Section 27.2.6.5](#) for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

#### **27.5.50.14 Good Transmit Frames Register (TXGOODFRAMES)**

The total number of good frames transmitted on the EMAC. A good frame is defined as having all of the following:

- Any data or MAC control frame that was destined for any unicast, broadcast, or multicast address
- Was any length
- Had no late or excessive collisions, no carrier loss, and no underrun

**27.5.50.15 Broadcast Transmit Frames Register (TXBCASTFRAMES)**

The total number of good broadcast frames transmitted on the EMAC. A good broadcast frame is defined as having all of the following:

- Any data or MAC control frame destined for address FF-FF-FF-FF-FF-FFh only
- Was of any length
- Had no late or excessive collisions, no carrier loss, and no underrun

**27.5.50.16 Multicast Transmit Frames Register (TXMCASTFRAMES)**

The total number of good multicast frames transmitted on the EMAC. A good multicast frame is defined as having all of the following:

- Any data or MAC control frame destined for any multicast address other than FF-FF-FF-FF-FF-FFh
- Was of any length
- Had no late or excessive collisions, no carrier loss, and no underrun

**27.5.50.17 Pause Transmit Frames Register (TXPAUSEFRAMES)**

The total number of IEEE 802.3X pause frames transmitted by the EMAC. Pause frames cannot underrun or contain a CRC error because they are created in the transmitting MAC, so these error conditions have no effect on this statistic. Pause frames sent by software are not included in this count. Since pause frames are only transmitted in full-duplex mode, carrier loss and collisions have no effect on this statistic.

Transmitted pause frames are always 64-byte multicast frames so appear in the multicast transmit frames register and 64 octet frames register statistics.

**27.5.50.18 Deferred Transmit Frames Register (TXDEFERRED)**

The total number of frames transmitted on the EMAC that first experienced deferment. Such a frame is defined as having all of the following:

- Was any data or MAC control frame destined for any unicast, broadcast, or multicast address
- Was any size
- Had no carrier loss and no underrun
- Experienced no collisions before being successfully transmitted
- Found the medium busy when transmission was first attempted, so had to wait.

CRC errors have no effect on this statistic.

**27.5.50.19 Transmit Collision Frames Register (TXCOLLISION)**

The total number of times that the EMAC experienced a collision. Collisions occur under two circumstances:

- When a transmit data or MAC control frame has all of the following:
  - Was destined for any unicast, broadcast, or multicast address
  - Was any size
  - Had no carrier loss and no underrun
  - Experienced a collision. A jam sequence is sent for every non-late collision, so this statistic increments on each occasion if a frame experiences multiple collisions (and increments on late collisions).
- When the EMAC is in half-duplex mode, flow control is active, and a frame reception begins.

CRC errors have no effect on this statistic.

#### **27.5.50.20 Transmit Single Collision Frames Register (TXSINGLECOLL)**

The total number of frames transmitted on the EMAC that experienced exactly one collision. Such a frame is defined as having all of the following:

- Was any data or MAC control frame destined for any unicast, broadcast, or multicast address
- Was any size
- Had no carrier loss and no underrun
- Experienced one collision before successful transmission. The collision was not late.

CRC errors have no effect on this statistic.

#### **27.5.50.21 Transmit Multiple Collision Frames Register (TXMULTICOLL)**

The total number of frames transmitted on the EMAC that experienced multiple collisions. Such a frame is defined as having all of the following:

- Was any data or MAC control frame destined for any unicast, broadcast, or multicast address
- Was any size
- Had no carrier loss and no underrun
- Experienced 2 to 15 collisions before being successfully transmitted. None of the collisions were late.

CRC errors have no effect on this statistic.

#### **27.5.50.22 Transmit Excessive Collision Frames Register (TXEXCESSIVECOLL)**

The total number of frames when transmission was abandoned due to excessive collisions. Such a frame is defined as having all of the following:

- Was any data or MAC control frame destined for any unicast, broadcast, or multicast address
- Was any size
- Had no carrier loss and no underrun
- Experienced 16 collisions before abandoning all attempts at transmitting the frame. None of the collisions were late.

CRC errors have no effect on this statistic.

#### **27.5.50.23 Transmit Late Collision Frames Register (TXLATECOLL)**

The total number of frames when transmission was abandoned due to a late collision. Such a frame is defined as having all of the following:

- Was any data or MAC control frame destined for any unicast, broadcast, or multicast address
- Was any size
- Had no carrier loss and no underrun
- Experienced a collision later than 512 bit-times into the transmission. There may have been up to 15 previous (non-late) collisions that had previously required the transmission to be reattempted. The late collisions statistic dominates over the single, multiple, and excessive collisions statistics. If a late collision occurs, the frame is not counted in any of these other three statistics.

CRC errors, carrier loss, and underrun have no effect on this statistic.

#### **27.5.50.24 Transmit Underrun Error Register (TXUNDERRUN)**

The number of frames sent by the EMAC that experienced FIFO underrun. Late collisions, CRC errors, carrier loss, and underrun have no effect on this statistic.

**27.5.50.25 Transmit Carrier Sense Errors Register (TXCARRIERSENSE)**

The total number of frames on the EMAC that experienced carrier loss. Such a frame is defined as having all of the following:

- Was any data or MAC control frame destined for any unicast, broadcast, or multicast address
- Was any size
- The carrier sense condition was lost or never asserted when transmitting the frame (the frame is not retransmitted)

CRC errors and underrun have no effect on this statistic.

**27.5.50.26 Transmit Octet Frames Register (TXOCTETS)**

The total number of bytes in all good frames transmitted on the EMAC. A good frame is defined as having all of the following:

- Any data or MAC control frame that was destined for any unicast, broadcast, or multicast address
- Was any length
- Had no late or excessive collisions, no carrier loss, and no underrun

**27.5.50.27 Transmit and Receive 64 Octet Frames Register (FRAME64)**

The total number of 64-byte frames received and transmitted on the EMAC. Such a frame is defined as having all of the following:

- Any data or MAC control frame that was destined for any unicast, broadcast, or multicast address
- Did not experience late collisions, excessive collisions, underrun, or carrier sense error
- Was exactly 64-bytes long. (If the frame was being transmitted and experienced carrier loss that resulted in a frame of this size being transmitted, then the frame is recorded in this statistic).

CRC errors, alignment/code errors, and overruns do not affect the recording of frames in this statistic.

**27.5.50.28 Transmit and Receive 65 to 127 Octet Frames Register (FRAME65T127)**

The total number of 65-byte to 127-byte frames received and transmitted on the EMAC. Such a frame is defined as having all of the following:

- Any data or MAC control frame that was destined for any unicast, broadcast, or multicast address
- Did not experience late collisions, excessive collisions, underrun, or carrier sense error
- Was 65-bytes to 127-bytes long

CRC errors, alignment/code errors, underruns, and overruns do not affect the recording of frames in this statistic.

**27.5.50.29 Transmit and Receive 128 to 255 Octet Frames Register (FRAME128T255)**

The total number of 128-byte to 255-byte frames received and transmitted on the EMAC. Such a frame is defined as having all of the following:

- Any data or MAC control frame that was destined for any unicast, broadcast, or multicast address
- Did not experience late collisions, excessive collisions, underrun, or carrier sense error
- Was 128-bytes to 255-bytes long

CRC errors, alignment/code errors, underruns, and overruns do not affect the recording of frames in this statistic.



**27.5.50.30 Transmit and Receive 256 to 511 Octet Frames Register (FRAME256T511)**

The total number of 256-byte to 511-byte frames received and transmitted on the EMAC. Such a frame is defined as having all of the following:

- Any data or MAC control frame that was destined for any unicast, broadcast, or multicast address
- Did not experience late collisions, excessive collisions, underrun, or carrier sense error
- Was 256-bytes to 511-bytes long

CRC errors, alignment/code errors, underruns, and overruns do not affect the recording of frames in this statistic.

**27.5.50.31 Transmit and Receive 512 to 1023 Octet Frames Register (FRAME512T1023)**

The total number of 512-byte to 1023-byte frames received and transmitted on the EMAC. Such a frame is defined as having all of the following:

- Any data or MAC control frame that was destined for any unicast, broadcast, or multicast address
- Did not experience late collisions, excessive collisions, underrun, or carrier sense error
- Was 512-bytes to 1023-bytes long

CRC errors, alignment/code errors, and overruns do not affect the recording of frames in this statistic.

**27.5.50.32 Transmit and Receive 1024 to RXMAXLEN Octet Frames Register (FRAME1024TUP)**

The total number of 1024-byte to RXMAXLEN-byte frames received and transmitted on the EMAC. Such a frame is defined as having all of the following:

- Any data or MAC control frame that was destined for any unicast, broadcast, or multicast address
- Did not experience late collisions, excessive collisions, underrun, or carrier sense error
- Was 1024-bytes to RXMAXLEN-bytes long

CRC/alignment/code errors, underruns, and overruns do not affect frame recording in this statistic.

**27.5.50.33 Network Octet Frames Register (NETOCTETS)**

The total number of bytes of frame data received and transmitted on the EMAC. Each frame counted has all of the following:

- Was any data or MAC control frame destined for any unicast, broadcast, or multicast address (address match does not matter)
- Was of any size (including less than 64-byte and greater than RXMAXLEN-byte frames)

Also counted in this statistic is:

- Every byte transmitted before a carrier-loss was experienced
- Every byte transmitted before each collision was experienced (multiple retries are counted each time)
- Every byte received if the EMAC is in half-duplex mode until a jam sequence was transmitted to initiate flow control. (The jam sequence is not counted to prevent double-counting).

Error conditions such as alignment errors, CRC errors, code errors, overruns, and underruns do not affect the recording of bytes in this statistic. The objective of this statistic is to give a reasonable indication of Ethernet utilization.



**27.5.50.34 Receive FIFO or DMA Start of Frame Overruns Register (RXSOFOVERRUNS)**

The total number of frames received on the EMAC that had either a FIFO or DMA start of frame (SOF) overrun. An SOF overrun frame is defined as having all of the following:

- Was any data or MAC control frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- Was of any size (including less than 64-byte and greater than RXMAXLEN-byte frames)
- The EMAC was unable to receive it because it did not have the resources to receive it (cell FIFO full or no DMA buffer available at the start of the frame).

CRC errors, alignment errors, and code errors have no effect on this statistic.

**27.5.50.35 Receive FIFO or DMA Middle of Frame Overruns Register (RXMOFOVERRUNS)**

The total number of frames received on the EMAC that had either a FIFO or DMA middle of frame (MOF) overrun. An MOF overrun frame is defined as having all of the following:

- Was any data or MAC control frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- Was of any size (including less than 64-byte and greater than RXMAXLEN-byte frames)
- The EMAC was unable to receive it because it did not have the resources to receive it (cell FIFO full or no DMA buffer available after the frame was successfully started - no SOF overrun).

CRC errors, alignment errors, and code errors have no effect on this statistic.

**27.5.50.36 Receive DMA Overruns Register (RXDMAOVERRUNS)**

The total number of frames received on the EMAC that had either a DMA start of frame (SOF) overrun or a DMA middle of frame (MOF) overrun. A receive DMA overrun frame is defined as having all of the following:

- Was any data or MAC control frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- Was of any size (including less than 64-byte and greater than RXMAXLEN-byte frames)
- The EMAC was unable to receive it because it did not have the DMA buffer resources to receive it (zero head descriptor pointer at the start or during the middle of the frame reception).

CRC errors, alignment errors, and code errors have no effect on this statistic.

## ***Universal Serial Bus (USB)***

---

---

---

This chapter describes the universal serial bus (USB) controller on the microcontroller.

<b>Topic</b>	<b>Page</b>
<b>28.1 Overview .....</b>	<b>1696</b>
<b>28.2 USB Host Controller .....</b>	<b>1697</b>
<b>28.3 USB Device Controller .....</b>	<b>1723</b>
<b>28.4 USB Connectivity .....</b>	<b>1808</b>

## 28.1 Overview

The microcontroller provides several varieties of USB functionality, including:

- USB host: device provides a two-port *USB Specification Revision 1.1*-compliant host controller, which is based on the *OHCI Specification for USB Release 1.0a*
- USB device: device provides a full-speed USB device
- Support for up to 2 active simultaneous ports (1 Function and 1 Host, or 2 Hosts)

## **28.2 USB Host Controller**

The USB host controller (HC) is a two-port controller that communicates with USB devices at the USB low-speed (1.5M bit-per-second maximum) and full-speed (12M bit-per-second maximum) data rates. It is compatible with the *Universal Serial Bus Specification Revision 2.0* and the *Open HCI—Open Host Controller Interface Specification for USB*, Release 1.0a, available through the Compaq Computer Corporation web site, and hereafter called the *OHCI Specification for USB*. It is assumed that users of the USB host controller are already familiar with the *USB Specification* and *OHCI Specification for USB*.

The USB host controller implements the register set and makes use of the memory data structures defined in the *OHCI Specification for USB*. These registers and data structures are the mechanism by which a USB host controller driver software package can control the USB host controller. The *OHCI Specification for USB* also defines how the USB host controller implementation must interact with those registers and data structures in system memory. The device CPU accesses these registers via the module's peripheral bus.

To reduce processor software and interrupt overhead, the USB host controller generates USB traffic based on data structures and data buffers stored in system memory. The USB host controller accesses these data structures without direct intervention by the processor using the master access (bus transaction initiator) port. These data structures and data buffers can be located in internal or external RAM.

The USB host controller provides an interrupt to the Vectored Interrupt Manager (VIM) to signal certain hardware events to the host controller driver software.

### **28.2.1 USB Open Host Controller Interface Functionality**

#### **28.2.1.1 OHCI Controller Overview**

The *Open HCI—Open Host Controller Interface Specification for USB*, Release 1.0a, defines a set of registers and data structures stored in system memory that define how a USB host controller interfaces to system software. This specification, in conjunction with the *Universal Serial Bus Specification Version 2.0*, defines most of the USB functionality that the USB host controller provides.

The *OHCI Specification for USB* focuses on two main aspects of the hardware implementation of a USB host controller: its register set and the memory data structures that define the activity to appear on the USB bus. Also discussed are issues such as interrupt generation, USB host controller state, USB frame management, and the methods that the hardware must use to process the lists of data structures in system memory.

This document does not duplicate the information presented in the *OHCI Specification for USB* or the *USB Specification*. USB host controller users can refer to the *USB Specification* and the *OHCI Specification for USB* for detailed discussions of USB requirements and OHCI controller operation.

### **28.2.2 USB Host Controller Differences From OHCI Specification for USB**

The USB host controller implementation does not implement every aspect of the functionality defined in the *OHCI Specification for USB*. The differences focus on the OHCI ownership change interrupt. Other restrictions are imposed by the effects of the device pin multiplexing options.

#### **28.2.2.1 Top-Level Pin Multiplexing and OHCI Registers**

The RM4x microcontroller supports two USB Host ports and one USB Device port. The USB device port interface signals are multiplexed with the second USB Host port interface signals. All USB interface signals are also multiplexed with other functions on the RM4x microcontrollers. The configuration of these multiplexors is done by configuring the I/O Multiplexing Module control registers. Refer to the I/O Multiplexing Module chapter of the Technical Reference Manual for more information on the sequence for configuring these control registers.

The OHCI RHDESCRIPTORA register always reports two available USB host ports, regardless of the top-level pin multiplexing settings.

**Table 28-1. USB Host / Device Interface Signal Multiplexing and Control**

USB Host / Device Interface Signal Name	Control for Selecting USB Host Signal	Control for Selecting USB Device Signal
USB1.OVERCURRENT	PINMMR14 [7:0] = 0b00001000	-
USB1.RCV	PINMMR13 [31:24] = 0b00001000	-
USB1.VM	PINMMR12 [31:24] = 0b00001000	-
USB1.VP	PINMMR12 [23:16] = 0b00001000	-
USB1.PORTPOWER	PINMMR21 [15:8] = 0b00000010	-
USB1.SPEED	PINMMR19 [15:8] = 0b00000100	-
USB1.SUSPEND	PINMMR20 [23:16] = 0b00001000	-
USB1.TXDAT	PINMMR18 [31:24] = 0b00000010	-
USB1.TXEN	PINMMR17 [7:0] = 0b00000100	-
USB1.TXSE0	PINMMR18 [15:8] = 0b00000010	-
USB2.OVERCURRENT / USB_FUNC.VBUSI	PINMMR1 [15:8] = 0b00001000	PINMMR1 [15:8] = 0b00010000
USB2.RCV / USB_FUNC.RXDI	PINMMR0 [7:0] = 0b00000010	PINMMR0 [7:0] = 0b00000100
USB2.VM / USB_FUNC.RXDMI	PINMMR1 [7:0] = 0b00000010	PINMMR1 [7:0] = 0b00000100
USB2.VP / USB_FUNC.RXDPI	PINMMR0 [15:8] = 0b00000010	PINMMR0 [15:8] = 0b00000100
USB2.PORTPOWER / USB_FUNC.GZO	PINMMR6 [7:0] = 0b00000010	PINMMR6 [7:0] = 0b00000100
USB2.SPEED / USB_FUNC.PUENON	PINMMR4 [31:24] = 0b00000100	PINMMR4 [31:24] = 0b00001000
USB2.SUSPEND / USB_FUNC.SUSPENDO	PINMMR6 [23:16] = 0b00000100	PINMMR6 [23:16] = 0b00001000
USB2.TXDAT / USB_FUNC.TXDO	PINMMR2 [7:0] = 0b00000010	PINMMR2 [7:0] = 0b00000100
USB2.TXEN / USB_FUNC.PUENO	PINMMR4 [23:16] = 0b00000100	PINMMR4 [23:16] = 0b00001000
USB2.TXSE0 / USB_FUNC.SE0O	PINMMR3 [15:8] = 0b00000010	PINMMR3 [15:8] = 0b00000100

### 28.2.2.2 No Ownership Change Interrupt

The USB host controller does not implement the OHCI ownership change interrupt.

### 28.2.3 Implementation of OHCI Specification for USB

#### 28.2.3.1 Isochronous Transmit Descriptor (TD) OFFSETX/PSWX Values

The USB host controller implements the *OHCI Specification for USB* optional feature of checking isochronous OFFSETX/PSWX values. If either OFFSETX or OFFSET(X+1) does not have a condition code of *Not Accessed*, or if the OFFSET(X+1) value is not greater than or equal to OFFSETX, then an unrecoverable error is reported. Unrecoverable errors issued for these reasons do not cause an update of the HOSTUEADDR, HOSTUESTATUS, or HOSTTIMEOUTCTRL registers.

#### 28.2.3.2 USB Host Controller Endpoint Descriptor (ED) List Head Pointers

The *OHCI Specification for USB* provides a specific sequence of operations for the host controller driver to perform when setting up the host controller. Failure to follow that sequence can result in malfunction. As a specific example, the HCCONTROLHEADED and HCBULKHEADED pointer registers and the 32 HCCAINERRUPTABLE pointers must all point to valid physical addresses of valid endpoint descriptors.

The USB host controller does not check HCCONTROLHEADED registers, HCBULKHEADED registers, or the values in the 32 HCCAINERRUPTABLE pointers before using them to access EDs. If any of these pointers are NULL when the corresponding list enable bit is set, the USB host controller attempts to access using the physical address of 0, which causes an unrecoverable error to be signaled. HOSTUEADDR, HOSTUESTATUS, and HOSTTIMEOUTCTRL registers are updated in this case.

#### 28.2.3.3 OHCI USB Suspend State

The USB host controller ignores upstream traffic from downstream devices for about 3 ms after the host

controller state (HCCONTROL.HCFS) changes from USB resume state to USB operational state. If any TDs cause generation of downstream packets during that time, the downstream packets are sent, but any response provided by the downstream device is ignored. Any such TDs are aborted with completion codes marked as *Device Not Responding*. TDs on any of the lists (periodic, control, bulk, and isochronous) can cause such an occurrence.

The USB specification requires that system software must provide a 10-ms resume recovery time ( $T_{RSMRCY}$ ) after a bus segment transitions from resume signaling to normal operational mode. During that time, only start of frame packets are to be sent on the bus segment. It is recommended that system software disable all list enable bits (HCCONTROL.PLE, HCCONTROL.IE, HCCONTROL.CLE, and HCCONTROL.BLE) and then wait for at least 1 ms before setting the host controller into USB suspend state (via HCCONTROL.HCFS). When restoring from suspend, system software must set the host controller into USB resume state, and wait for the host controller to transition into USB operational state. System software must then wait 10 ms before enabling the host controller list enable bits.

When the host controller has been placed into the USB suspend state under software control, but is brought out by a remote wake-up, system software must monitor the HCRHPORTSTATUS[x].PSS and HCRHPORTSTATUS[x].PSSC bits. The HCRHPORTSTATUS[x].PSS bit changes to 0 only after completion of resume signaling on the bus segment completes and completion of the 3-ms period where packets from downstream devices are ignored.

When using port-specific suspend, it is not necessary to disable the host controller lists so long as there are no active EDs and TDs directed toward devices that are downstream of the suspended port. For port-specific suspend operations, the host controller does not issue a root hub status change interrupt with the HCRHPORTSTATUS[n].PSSC bit = 1 and HCRHPORTSTATUS[n].PSS = 0 until after the approximately 3-ms delay after resume signaling completes.

When using port-specific suspend, system software must ensure that there are no active EDs for devices that are downstream of the suspended port before setting the port into suspend mode. While the port is in suspend or being resumed, system software must not enable any EDs for any devices downstream of the suspended port. Once the root hub status change interrupt occurs as a result of the suspended port PSS bit changing to 0, EDs can be enabled for devices downstream of the port that is now operational.

#### 28.2.4 USB Host Controller Registers

Most of the host controller (HC) registers are the OHCI operational registers, which are defined by the *OHCI Specification for USB*. Four additional registers not specified by the *OHCI Specification for USB* provide additional information about the USB host controller state. USB host controller registers can be accessed in user and supervisor modes.

The USB host controller registers are listed in [Table 28-2](#). [Section 28.2.4.1](#) through [Section 28.2.4.27](#) describe specific register bits. The base address for the USB Host controller registers is 0xFCF78B00.

**Table 28-2. USB Host Controller Registers**

Name	Description	R/W	Size <sup>(1)</sup>	Address
HCREVISION	OHCI revision number	R	32	FCF78B00h
HCCONTROL	HC operating mode	R/W	32	FCF78B04h
HCCOMMANDSTATUS	HC command and status	R/W	32	FCF78B08h
HCINTERRUPTSTATUS	HC interrupt status	R/W	32	FCF78B0Ch
HCINTERRUPTENABLE	HC interrupt enable	R/W	32	FCF78B10h
HCINTERRUPTDISABLE	HC interrupt disable	R/W	32	FCF78B14h
HCHCCA	Physical address of HCCA <sup>(2)</sup>	R/W	32	FCF78B18h
HCPERIODCURRENTED	Physical address of current periodic endpoint descriptor <sup>(2)</sup>	R/W	32	FCF78B1Ch
HCCONTROLHEADED	Physical address of head of control endpoint descriptor list <sup>(2)</sup>	R/W	32	FCF78B20h
HCCONTROLCURRENTED	Physical address of current control endpoint descriptor <sup>(2)</sup>	R/W	32	FCF78B24h

<sup>(1)</sup> Access to these registers must be by 32-bit reads or 32-bit writes. Use of other access sizes may result in undefined operation.

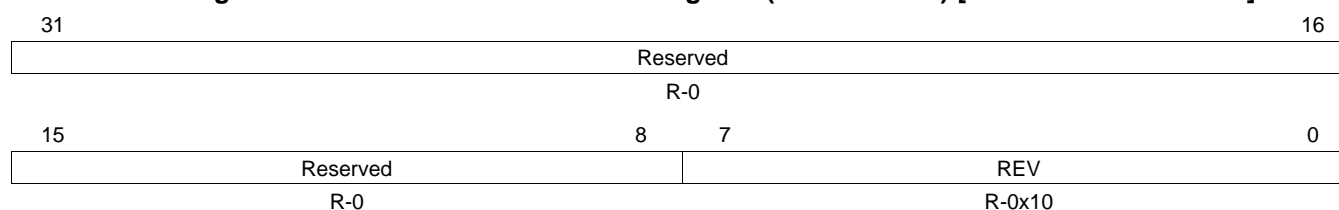
<sup>(2)</sup> Restrictions apply to the physical addresses used in these registers. See [Section 28.2.9](#), *Physical Addressing*.

**Table 28-2. USB Host Controller Registers (continued)**

Name	Description	R/W	Size <sup>(1)</sup>	Address
HCBULKHEADED	Physical address of head of bulk endpoint descriptor list <sup>(2)</sup>	R/W	32	FCF78B28h
HCBULKCURRENTED	Physical of current bulk endpoint descriptor <sup>(2)</sup>	R/W	32	FCF78B2Ch
HCDONEHEAD	Physical address of head of list of retired transfer descriptors <sup>(2)</sup>	R	32	FCF78B30h
HCFMINTERVAL	HC frame interval	R/W	32	FCF78B34h
HCFMREMAINING	HC frame remaining	R	32	FCF78B38h
HCFMNUMBER	HC frame number	R	32	FCF78B3Ch
HCPERIODICSTART	HC periodic start	R/W	32	FCF78B40h
HCLSTHRESHOLD	HC low speed threshold	R/W	32	FCF78B44h
HCRHDESCRIPTORA	HC root hub A	R/W	32	FCF78B48h
HCRHDESCRIPTORB	HC root hub B	R/W	32	FCF78B4Ch
HCRHSTATUS	HC root hub status	R/W	32	FCF78B50h
HCRHPORTSTATUS0	HC port 0 control and status	R/W	32	FCF78B54h
HCRHPORTSTATUS1	HC port 1 control and status	R/W	32	FCF78B58h
Reserved	Reserved	R/W	32	FCF78B5Ch
HOSTUEADDR	Host UE address	R	32	FCF78BE0h
HOSTUESTATUS	Host UE status	R	32	FCF78BE4h
HOSTTIMEOUTCTRL	Host time-out control	R/W	32	FCF78BE8h
HOSTREVISION	Host revision	R	32	FCF78BECh

**28.2.4.1 OHCI Revision Number Register (HCREVISION)**

The OHCI revision number (Figure 28-1) register reports the revision number of the *OHCI Specification for USB* upon which the USB host controller is based.

**Figure 28-1. OHCI Revision Number Register (HCREVISION) [address = FCF78B00h]**

LEGEND: R = Read, W = Write, n = value at reset

**Table 28-3. OHCI Revision Number Register (HCREVISION) Bit Field Descriptions**

Bit	Field	Value	Description
31 - 8	Reserved		Reserved
7 - 0	REV		OHCI specification revision. The OHCI revision number upon which the USB host controller is based. Write has no effect.

**28.2.4.2 HC Operating Mode Register (HCCONTROL)**

The HC operating mode register (Figure 28-2) controls the operating mode of the USB host controller.

**Figure 28-2. HC Operating Mode Register (HCCONTROL) [address = FCF78B04h]**

31	Reserved				16	
R-0						
15	11	10	9	8		
Reserved		RWE	RWC	IR		
R-0		R/W-0	R/W-0	R/W-0		
7	6	5	4	3	2	
HCFS		BLE	CLE	IE	PLE	
R/W-00		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 28-4. HC Operating Mode Register (HCCONTROL) Bit Field Descriptions**

Bit	Field	Value	Description
31 - 11	Reserved		Reserved
10	RWE		Remote wake-up enable. This bit has no effect in the device. The USB host controller does not provide a processor wake-up mechanism.
9	RWC		Remote wake-up connected. This bit has no effect in the device. The USB host controller does not provide a processor wake-up mechanism.
8	IR		Interrupt routing. The USB host controller does not provide an SMI interrupt. This bit must be 0 to allow the USB host controller interrupt to propagate to the Vectored Interrupt Manager (VIM).
7 - 6	HCFS	00 01 10 11	Host controller functional state: 00 USB reset 01 USB resume 10 USB operational 11 USB suspend  A transition to USB operational causes SOF generation to begin in 1 ms. The USB host controller can automatically transition from USB suspend to USB resume if a downstream resume is received. The USB host controller enters USB suspend after a software reset. The USB host controller enters USB reset after a hardware reset. The USB reset state resets the root hub and causes downstream signaling of USB reset.
5	BLE	0 1	Bulk list enable: 0 Bulk ED list not processed in the next 1-ms frame. Host controller driver can modify the list. If driver removes the ED pointed to by the HCBULKCURRENTED from the ED list, it must update HCBULKCURRENTED to point to an ED still on the list before it re-enables the bulk list. 1 Enables processing of bulk ED list. HCBULKHEADED must be 0 or point to a valid ED before setting this bit. HCBULKCURRENTED must point to a valid ED or be 0 before setting this bit.
4	CLE	0 1	Control list enable: 0 Control ED list is not processed in the next 1-ms frame. Host controller driver can modify the control ED list. If driver removes the ED pointed to by the HCCONTROLCURRENTED from the ED list, it must update HCCONTROLCURRENTED to point to an ED still on the list before it re-enables the control list. 1 Enables processing of the control ED list. HCCONTROLHEADED must be 0 or point to a valid ED before setting this bit. HCCONTROLCURRENTED must be 0 or point to a valid ED before setting this bit.
3	IE	0 1	Isochronous enable 0 Isochronous EDs are not processed. The USB host controller checks this bit every time it finds an isochronous ED in the periodic list. 1 Enables processing of isochronous EDs. When this bit is written to 1, processing of isochronous EDs can be enabled in the next frame, if not in the current frame.
2	PLE	0 1	Periodic list enable 0 The periodic ED lists are not processed. When written to 0, periodic list processing is disabled beginning with the next frame. 1 Enables processing of the periodic ED lists. When written to 1, periodic list processing begins in the next frame.



**Table 28-4. HC Operating Mode Register (HCCONTROL) Bit Field Descriptions (continued)**

Bit	Field	Value	Description
1 - 0	CBSR	00 01 10 11	Control/bulk service ratio. Specifies the ratio between control and bulk EDs processed in a frame. One control ED per bulk ED Two control EDs per bulk ED Three control EDs per bulk ED Four control EDs per bulk ED

**28.2.4.3 HC Command and Status Register (HCCOMMANDSTATUS)**

The HC command and status register shows the current state of the host controller and accepts commands from the host controller driver.

**Figure 28-3. HC Command and Status Register (HCCOMMANDSTATUS) [address = FCF78B08h]**

31	Reserved	18	17	16
	R-0		SOC R-00	
15	Reserved	4	3	2
	R-0		OCR	BLF
			CLF	HCR
			R/W-0	R/W-0
			R/W-0	R/W-0

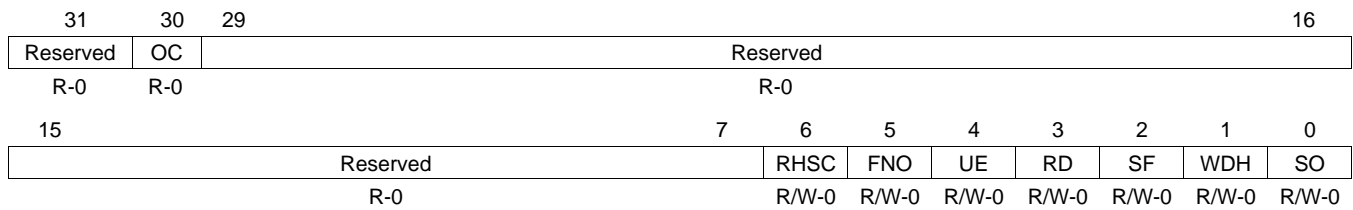
LEGEND: R = Read, W = Write, n = value at reset

**Table 28-5. HC Command and Status Register (HCCOMMANDSTATUS) Bit Field Descriptions**

Bit	Field	Value	Description
31 - 18	Reserved		Reserved
17 - 16	SOC		Scheduling overrun count Counts the number of times a scheduling overrun occurs. This count is incremented even if the host controller driver has not acknowledged any previous pending scheduling overrun interrupt.
15 - 4	Reserved		Reserved
3	OCR		Ownership change request This bit is set by the host controller driver to gain ownership of the host controller. The device does not support SMI interrupts, so no ownership change interrupt occurs.
2	BLF		Bulk list filled The host controller driver must set this bit if it modifies the bulk list to include new TDs. If HCBULKCURRENTED is 0, the USB host controller does not begin processing bulk list EDs unless this bit is set. When the USB host controller sees this bit set and begins processing the bulk list, it clears this bit.
1	CLF		Control list filled The host controller driver must set this bit if it modifies the control list to include new TDs. If HCCONTROLHEADED is 0, the USB host controller does not begin processing control list EDs unless this bit is set. When the USB host controller sees this bit set and begins processing the control list, it clears this bit.
0	HCR		Host controller reset. Write of 0 has no effect. Value of 1 initiates a software reset of the USB host controller. This transitions the USB host controller to the USB suspend state. This resets most USB host controller OHCI registers. OHCI register accesses must not be attempted until a read of this bit returns a 0. A write of 1 to this bit does not reset the root hub and does not signal USB reset to downstream USB functions.

**28.2.4.4 HC Interrupt Status Register (HCINTERRUPTSTATUS)**

The HC interrupt status register reports the status of the USB host controller internal interrupt sources.

**Figure 28-4. HC Interrupt Status Register (HCINTERRUPTSTATUS) [address = FCF78B0Ch]**


LEGEND: R = Read, W = Write, n = value at reset

**Table 28-6. HC Interrupt Status Register (HCINTERRUPTSTATUS) Bit Field Descriptions**

Bit	Field	Value	Description
31	Reserved		Reserved
30	OC		Ownership change The USB host controller does not implement ownership change interrupts.
29 - 7	Reserved		Reserved
6	RHSC	0 1	Root hub status change When 1, indicates a root hub status change has occurred. Write of 0 has no effect. Write of 1 clears this bit.
5	FNO	0 1	Frame number overflow When 1, indicates a frame number overflow has occurred. Write of 0 has no effect. Write of 1 clears this bit.
4	UE	0 1	Unrecoverable error When 1, indicates that an unrecoverable error has occurred on the OCPI bus or that an isochronous TD PSW field condition code was not set to <i>Not Accessed</i> when the USB host controller attempted to perform a transfer using that PSW/offset pair. Write of 0 has no effect. Write of 1 clears this bit.
3	RD	0 1	Resume detected When 1, indicates that a downstream device has issued a resume request. Write of 0 has no effect. Write of 1 clears this bit.
2	SF	0 1	Start of frame When 1, indicates that a SOF has been issued. Write of 0 has no effect. Write of 1 clears this bit.
1	WDH	0 1	Write done head When 1, indicates that the USB host controller has updated the HCDONEHEAD register. Write of 0 has no effect. Write of 1 clears this bit. The host controller driver must read the value from HCDONEHEAD before writing 1 to this bit.
0	SO	0 1	Scheduling overrun When 1, indicates that a scheduling overrun has occurred. Write of 0 has no effect. Write of 1 clears this bit.

#### 28.2.4.5 HC Interrupt Enable Register (HCINTERRUPTENABLE)

The HC interrupt enable register ([Figure 28-5](#)) enables various OHCI interrupt sources to generate interrupts to the device level 2 interrupt handler.

**Figure 28-5. HC Interrupt Enable Register (HCINTERRUPTENABLE) [address = FCF78B10h]**

31	30	29									16			
MIE	OC	Reserved												
R/W-0	R-0	R-0												
15	Reserved						7	6	5	4	3	2	1	0
R-0							RHSC	FNO	UE	RD	SF	WDH	SO	
R-0							R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R = Read, W = Write, n = value at reset

**Table 28-7. HC Interrupt Enable Register (HCINTERRUPTENABLE) Bit Field Descriptions**

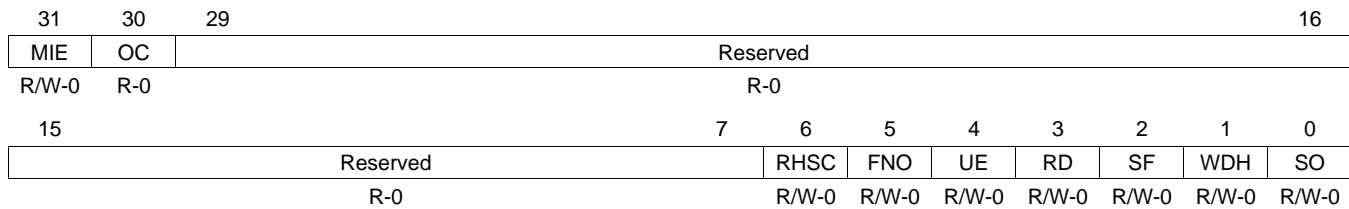
Bit	Field	Value	Description
31	MIE	0 1	<p>Master interrupt enable</p> <p>When 1, allows other enabled OHCI interrupt sources to propagate to the Vectored Interrupt Manager (VIM).</p> <p>When 0, OHCI interrupt sources are ignored and no USB host controller interrupts are propagated to the Vectored Interrupt Manager (VIM).</p> <p>A write of 0 has no effect on this bit.</p> <p>A write of 1 sets this bit.</p>
30	OC		<p>Ownership change</p> <p>This bit has no effect on the device.</p>
29 - 7	Reserved		Reserved
6	RHSC	0 1	<p>Root hub status change</p> <p>When 1 and MIE is 1, allows root hub status change interrupts to propagate to the Vectored Interrupt Manager (VIM).</p> <p>When 0, or when MIE is 0, root hub status change interrupts do not propagate.</p> <p>A write of 0 has no effect on this bit</p> <p>.A write of 1 sets this bit.</p>
5	FNO	0 1	<p>Frame number overflow</p> <p>When 1 and MIE is 1, allows frame number overflow interrupts to propagate to the Vectored Interrupt Manager (VIM).</p> <p>When 0, or when MIE is 0, frame number overflow interrupts do not propagate.</p> <p>A write of 0 has no effect on this bit.</p> <p>A write of 1 sets this bit.</p>
4	UE	0 1	<p>Unrecoverable error</p> <p>When 1 and MIE is 1, allows unrecoverable error interrupts to propagate to the Vectored Interrupt Manager (VIM).</p> <p>When 0, or when MIE is 0, unrecoverable error interrupts do not propagate.</p> <p>A write of 0 has no effect on this bit.</p> <p>A write of 1 sets this bit.</p>
3	RD	0 1	<p>Resume detected</p> <p>When 1 and MIE is 1, allows resume detected interrupts to propagate to the Vectored Interrupt Manager (VIM).</p> <p>When 0, or when MIE is 0, resume detected interrupts do not propagate.</p> <p>A write of 0 has no effect on this bit.</p> <p>A write of 1 sets this bit.</p>
2	SF	0 1	<p>Start of frame</p> <p>When 1 and MIE is 1, allows start of frame interrupts to propagate to the Vectored Interrupt Manager (VIM).</p> <p>When 0, or when MIE is 0, start of frame interrupts do not propagate.</p> <p>A write of 0 has no effect on this bit.</p> <p>A write of 1 sets this bit.</p>

**Table 28-7. HC Interrupt Enable Register (HCINTERRUPTENABLE) Bit Field Descriptions (continued)**

Bit	Field	Value	Description
1	WDH	0 1	Write done head When 1 and MIE is 1, allows write done head interrupts to propagate to the Vectored Interrupt Manager (VIM). When 0, or when MIE is 0, write done head interrupts do not propagate. A write of 0 has no effect on this bit. A write of 1 sets this bit.
0	SO	0 1	Scheduling overrun When 1 and MIE is 1, allows scheduling overrun interrupts to propagate to the Vectored Interrupt Manager (VIM). When 0, or when MIE is 0, scheduling overrun interrupts do not propagate. A write of 0 has no effect on this bit. A write of 1 sets this bit.

**28.2.4.6 HC Interrupt Disable Register (HCINTERRUPTDISABLE)**

The HC interrupt disable register is used to clear bits in the HCINTERRUPTENABLE register.

**Figure 28-6. HC Interrupt Disable Register (HCINTERRUPTDISABLE) [address = FCF78B14h]**


LEGEND: R = Read, W = Write, n = value at reset

**Table 28-8. HC Interrupt Disable Register (HCINTERRUPTDISABLE) Bit Field Descriptions**

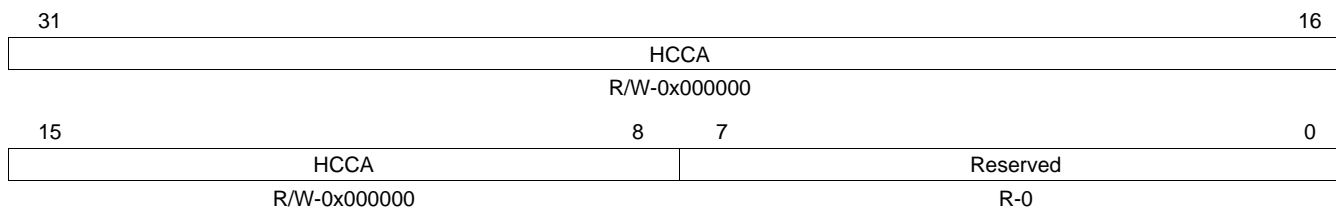
Bit	Field	Value	Description
31	MIE	0 1	Master interrupt enable Read always returns 0. Write of 0 has no effect. Write of 1 clears the HCINTERRUPTENABLE MIE bit.
30	OC		Ownership change This bit has no effect on the device.
29 - 7	Reserved		Reserved
6	RHSC	0 1	Root hub status change Read always returns 0. Write of 0 has no effect. Write of 1 clears the HCINTERRUPTENABLE RHSC bit.
5	FNO	0 1	Frame number overflow Read always returns 0. Write of 0 has no effect. Write of 1 clears the HCINTERRUPTENABLE FNO bit.
4	UE	0 1	Unrecoverable error Read always returns 0. Write of 0 has no effect. Write of 1 clears the HCINTERRUPTENABLE UE bit.

**Table 28-8. HC Interrupt Disable Register (HCINTERRUPTDISABLE) Bit Field Descriptions (continued)**

Bit	Field	Value	Description
3	RD	0 1	Resume detected Read always returns 0. Write of 0 has no effect. Write of 1 clears the HCINTERRUPTENABLE RD bit.
2	SF	0 1	Start of frame Read always returns 0. Write of 0 has no effect. Write of 1 clears the HCINTERRUPTENABLE SF bit.
1	WDH	0 1	Write done head Read always returns 0. Write of 0 has no effect. Write of 1 clears the HCINTERRUPTENABLE WDH bit.
0	SO	0 1	Scheduling overrun Read always returns 0. Write of 0 has no effect. Write of 1 clears the HCINTERRUPTENABLE SO bit.

**28.2.4.7 HC HCCA Address Register (HCHCCA)**

The HCCA address register defines the physical address of the beginning of the HCCA.

**Figure 28-7. HC HCCA Address Register (HCHCCA) [address = FCF78B18h]**

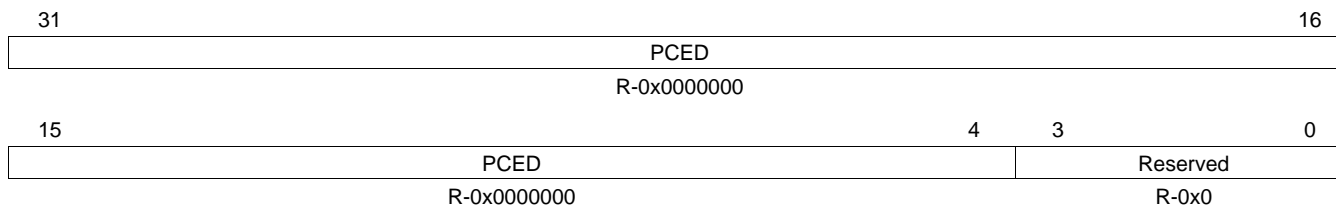
LEGEND: R = Read, W = Write, n = value at reset

**Table 28-9. HC HCCA Address Register (HCHCCA) Bit Field Descriptions**

Bit	Field	Value	Description
31 - 8	HCCA		Physical address of the beginning of the HCCA
7 - 0	Reserved		Reserved

**28.2.4.8 HC Current Periodic Register (HCPERIODCURRENTED)**

The HC current periodic register defines the physical address of the next endpoint descriptor (ED) on the periodic ED List.

**Figure 28-8. HC Current Periodic Register (HCPERIODCURRENTED) [address = FCF78B1Ch]**

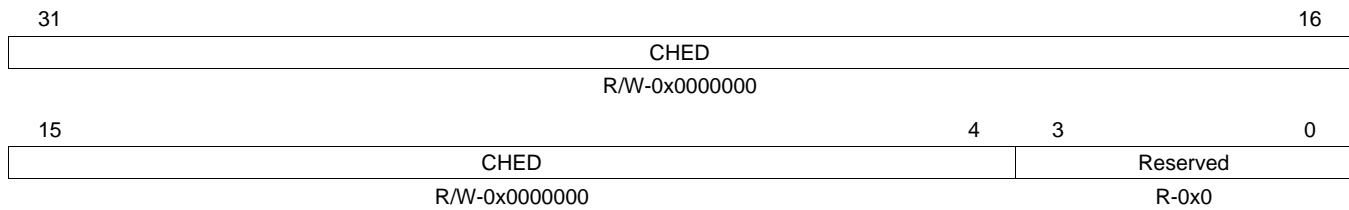
LEGEND: R = Read, W = Write, n = value at reset

**Table 28-10. HC Current Periodic Register (HCPERIODCURRENTED) Bit Field Descriptions**

Bit	Field	Value	Description
31 - 4	PCED		Physical address of current ED on the periodic ED list.  This field represents bits 31:4 of the physical address of the next ED on the periodic ED List. EDs are assumed to begin at 16-byte-aligned address, so bits 3:0 of this pointer are assumed to be 0. See <a href="#">Section 28.2.9</a> for the restrictions on physical addresses.
3 - 0	Reserved		Reserved

**28.2.4.9 HC Head Control Register (HCONTROLHEADED)**

The HC head control register defines the physical address of the head ED of the control ED list.

**Figure 28-9. HC Head Control Register (HCONTROLHEADED) [address = FCF78B20h]**


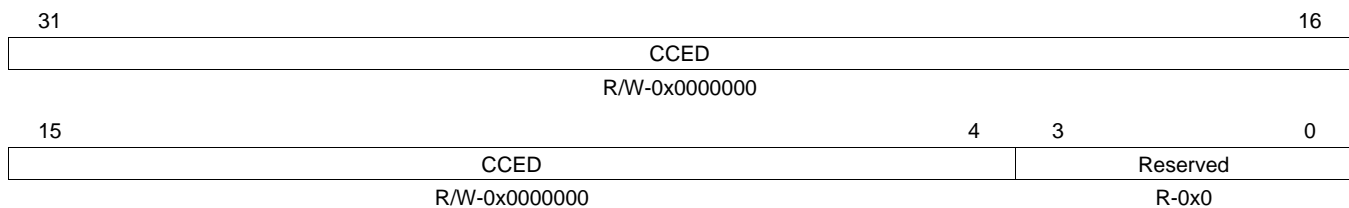
LEGEND: R = Read, W = Write, n = value at reset

**Table 28-11. HC Head Control Register (HCONTROLHEADED) Bit Field Descriptions**

Bit	Field	Value	Description
31 - 4	CHED		Physical address of current ED on the periodic ED list.  This field represents bits 31:4 of the physical address of the next ED on the periodic ED List. EDs are assumed to begin at 16-byte-aligned address, so bits 3:0 of this pointer are assumed to be 0. See <a href="#">Section 28.2.9</a> for the restrictions on physical addresses.
3 - 0	Reserved		Reserved

**28.2.4.10 HC Current Control Register (HCONTROLCURRENTED)**

The HC current control register defines the physical address of the next ED on the control ED list.

**Figure 28-10. HC Current Control Register (HCONTROLCURRENTED) [address = FCF78B24h]**


LEGEND: R = Read, W = Write, n = value at reset

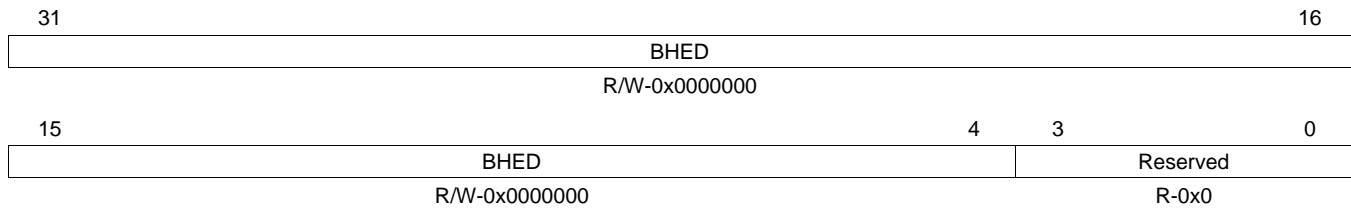
**Table 28-12. HC Current Control Register (HCONTROLCURRENTED) Bit Field Descriptions**

Bit	Field	Value	Description
31 - 4	CCED		Physical address of current ED on the periodic ED list.  This field represents bits 31:4 of the physical address of the next ED on the periodic ED List. EDs are assumed to begin at 16-byte-aligned address, so bits 3:0 of this pointer are assumed to be 0. See <a href="#">Section 28.2.9</a> for the restrictions on physical addresses.  A value of 0x00000000 indicates that the USB host controller has reached the end of the control ED list without finding any transfers to process.  This register is automatically updated by the USB host controller.
3 - 0	Reserved		Reserved

### 28.2.4.11 HC Head Bulk Register (HCBULKHEADED)

The head bulk register defines the physical address of the head ED on the bulk ED list.

**Figure 28-11. HC Head Bulk Register (HCBULKHEADED) [address = FCF78B28h]**



LEGEND: R = Read, W = Write, n = value at reset

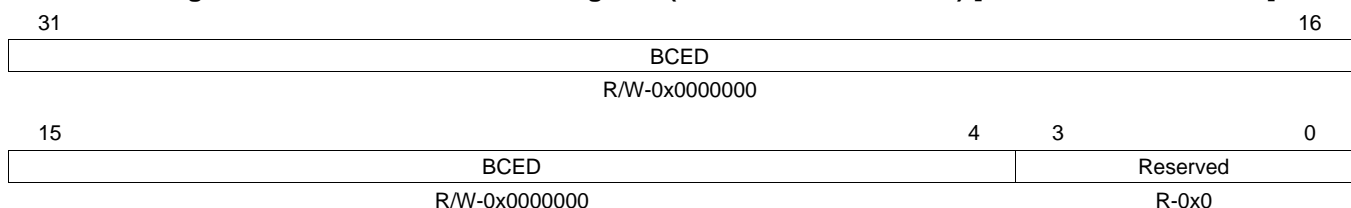
**Table 28-13. HC Head Bulk Register (HCBULKHEADED) Bit Field Descriptions**

Bit	Field	Value	Description
31 - 4	BHED		Physical address of current ED on the periodic ED list. This field represents bits 31:4 of the physical address of the next ED on the periodic ED List. EDs are assumed to begin at 16-byte-aligned address, so bits 3:0 of this pointer are assumed to be 0. See <a href="#">Section 28.2.9</a> for the restrictions on physical addresses. This register is automatically updated by the USB host controller.
3 - 0	Reserved		Reserved

### 28.2.4.12 HC Current Bulk Register (HCBULKCURRENTED)

The current bulk register defines the physical address of the next ED on the bulk ED list.

**Figure 28-12. HC Current Bulk Register (HCBULKCURRENTED) [address = FCF78B2Ch]**



LEGEND: R = Read, W = Write, n = value at reset

**Table 28-14. HC Current Bulk Register (HCBULKCURRENTED) Bit Field Descriptions**

Bit	Field	Value	Description
31 - 4	BCED		Physical address of current ED on the periodic ED list. This field represents bits 31:4 of the physical address of the next ED on the periodic ED List. EDs are assumed to begin at 16-byte-aligned address, so bits 3:0 of this pointer are assumed to be 0. See <a href="#">Section 28.2.9</a> for the restrictions on physical addresses. A value of 0x00000000 indicates that the USB host controller has reached the end of the bulk ED list without finding any transfers to process. This register is automatically updated by the USB host controller.
3 - 0	Reserved		Reserved

### 28.2.4.13 HC Head Done Register (HCDONEHEAD)

The head done register defines the physical address of the current head of the done TD queue.

**Figure 28-13. HC Head Done Register (HCDONEHEAD) [address = FCF78B30h]**

31	DH	16
R/W-0x0000000		
15	DH	0
R/W-0x0000000		Reserved R-0x0

LEGEND: R = Read, W = Write, n = value at reset

**Table 28-15. HC Head Done Register (HCDONEHEAD) Bit Field Descriptions**

Bit	Field	Value	Description
31 - 4	DH		Physical address of current ED on the periodic ED list.  This field represents bits 31:4 of the physical address of the next ED on the periodic ED List. EDs are assumed to begin at 16-byte-aligned address, so bits 3:0 of this pointer are assumed to be 0. See <a href="#">Section 28.2.9</a> for the restrictions on physical addresses.  This register is automatically updated by the USB host controller.
3 - 0	Reserved		Reserved

#### 28.2.4.14 HC Frame Interval Register (HCFMINTERVAL)

The frame interval register defines the number of 12-MHz clock pulses in each USB frame.

**Figure 28-14. HC Frame Interval Register (HCFMINTERVAL) [address = FCF78B34h]**

31	30	FIT	FSMPS	16
		R/W-0		R/W-0x0000
15	14	13	FI	0
Reserved		R-0		
		R/W-0x2EDF		

LEGEND: R = Read, W = Write, n = value at reset

**Table 28-16. HC Frame Interval Register (HCFMINTERVAL) Bit Field Descriptions**

Bit	Field	Value	Description
31	FIT		Frame interval toggle  The host controller driver must toggle this bit any time it changes the frame interval field.
30 - 16	FSMPS		Largest data packet  Largest data packet size allowed for full-speed packets, in bit times.
15 - 14	Reserved		Reserved
13 - 0	FI		Frame interval  Number of 12-MHz clocks in the USB frame. Nominally, this is set to 11,999 to give a 1-ms frame. The host controller driver can make minor changes to this field to attempt to manually synchronize with another clock source.

#### 28.2.4.15 HC Frame Remaining Register (HCFMREMAINING)

The HC frame remaining register reports the number of full-speed bit times remaining in the current frame.



**Figure 28-15. HC Frame Remaining Register (HCFMREMAINING) [address = FCF78B38h]**

31	30	16
FRT		Reserved
R-0		R-0
15	14	13
Reserved		FR
R-0		R-0x0000

LEGEND: R = Read, W = Write, n = value at reset

**Table 28-17. HC Frame Remaining Register (HCFMREMAINING) Bit Field Descriptions**

Bit	Field	Value	Description
31	FRT		Frame interval toggle The host controller driver must toggle this bit any time it changes the frame interval field.
30 - 14	Reserved		Reserved
13 - 0	FR		Frame remaining The number of full-speed bit times remaining in the current frame. This field is automatically reloaded with the frame interval field value at the beginning of every frame.

**28.2.4.16 HC Frame Number Register (HCFMNUMBER)**

The HC frame number register reports the current USB frame number.

**Figure 28-16. HC Frame Number Register (HCFMNUMBER) [address = FCF78B3Ch]**

31	16
Reserved	
R-0	
15	0
FN	
R-0x0000	

LEGEND: R = Read, W = Write, n = value at reset

**Table 28-18. HC Frame Number Register (HCFMNUMBER) Bit Field Descriptions**

Bit	Field	Value	Description
31 - 14	Reserved		Reserved
13 - 0	FN		Frame number This field reports the current USB frame number. It is incremented when the frame remaining field is reloaded with the frame interval field value. Frame number automatically rolls over from 0xFFFF to 0x0000. After frame number is incremented, its new value is written to the HCCA and the USB host controller sets the SOF interrupt status bit and begins processing the ED lists.

**28.2.4.17 HC Periodic Start Register (HCPERIODICSTART)**

The HC periodic start register defines the position within the USB frame where EDs on the periodic list have priority over EDs on the bulk and control lists.

**Figure 28-17. HC Periodic Start Register (HCPERIODICSTART) [address = FCF78B40h]**

31	Reserved			16
R-0				
15	14	13	0	
Reserved		PS		
R-0		R/W-0x0000		

LEGEND: R = Read, W = Write, n = value at reset

**Table 28-19. HC Periodic Start Register (HCPERIODICSTART) Bit Field Descriptions**

Bit	Field	Value	Description
31 - 14	Reserved		Reserved
13 - 0	PS		Periodic start The host controller driver must program this value to be about 10% less than the frame interval field value so that control and bulk EDs have priority for the first 10% of the frame; then periodic EDs have priority for the remaining 90% of the frame.

#### 28.2.4.18 HC Low-Speed Threshold Register (HCLSTHRESHOLD)

The HC low-speed threshold register defines the latest time in a frame that the USB host controller can begin a low-speed packet.

**Figure 28-18. HC Low-Speed Threshold Register (HCLSTHRESHOLD) [address = FCF78B44h]**

31	Reserved			16
R-0				
15	14	13	0	
Reserved		LST		
R-0		R/W-0x0628		

LEGEND: R = Read, W = Write, n = value at reset

**Table 28-20. HC Low-Speed Threshold Register (HCLSTHRESHOLD) Bit Field Descriptions**

Bit	Field	Value	Description
31 - 14	Reserved		Reserved
13 - 0	LST		Low-speed threshold This field defines the number of full-speed bit times in the frame after which the USB host controller cannot start an 8-byte low-speed packet. The USB host controller only begins a low-speed transaction if the frame remaining field is greater than the low-speed threshold. The host controller driver must set this field to a value that ensures that an 8-byte low-speed TD completes before the end of the frame. When set, the host controller driver must not change the value.

#### 28.2.4.19 HC Root Hub A Register (HCRHDESCRIPTORA)

The HC root hub A register defines several aspects of the USB host controller root hub functionality.

**Figure 28-19. HC Root Hub A Register (HCRHDESCRIPTORA) [address = FCF78B48h]**

31							24	23			16	
POTPG						Reserved						
R/W-0x02						R-0x00						
15	13	12	11	10	9	8	7					0
Reserved		NOCP	OCPM	DT	NPS	PSM	NDP					
R/W-0		R/W-1	R/W-0	R-0	R/W-1	R/W-0	R-0x02					

LEGEND: R = Read, W = Write, n = value at reset

**Table 28-21. HC Root Hub A Register (HCRHDESCRIPTORA) Field Descriptions**

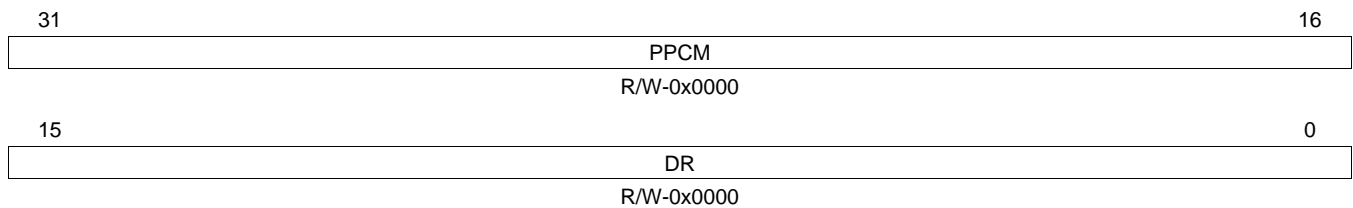
Bit	Field	Value	Description
31 - 24	POTPG		<p>Power-on to power-good time</p> <p>This field defines the minimum amount of time (2 ms × POTPG) between the USB host controller turning on power to a downstream port and when the USB host can access the downstream device. The default value is 0x2, so that the time is 4 ms.</p> <p>This field has no effect on USB host controller operation. After turning on power to a port, the USB host controller driver must delay the amount of time implied by POTPG before attempting to reset an attached downstream device.</p> <p>The required amount of time is system implementation-specific and must be calculated based on the amount of time the VBUS supply takes to provide valid VBUS to a worst-case downstream USB function controller.</p> <p>The implementation-specific value must be coCPUted and then written to this register before the USB host controller driver is initialized.</p> <p>Because the device does not provide a direct control from the USB host controller to switch VBUS on and off, this value must take into account any delays caused by other methods of controlling VBUS externally.</p> <p>System software can update this register to simplify host controller driver coding.</p>
23 - 13	Reserved		Reserved
12	NOCP	<p>0</p> <p>1</p>	<p>No overcurrent protection</p> <p>When 0, the USB Host ports support over-current protection inputs. The USB Host ports must not be in the Reset state for the over-current protection inputs to take effect.</p> <p>When 1, this bit indicates that the USB host controller does not implement overcurrent protection inputs. This bit defaults to 1 and can be cleared by the application software.</p>
11	OCPM	<p>0</p> <p>1</p>	<p>Overcurrent protection mode</p> <p>Over-current status is reported collectively for all downstream ports. This is the default value.</p> <p>Over-current status is reported on a per-port basis.</p> <p>This field is valid only if the NoOverCurrentProtection bit is cleared.</p>
10	DT	<p>0</p>	<p>Device type</p> <p>This bit is always 0, which indicates that the USB host controller implemented is not a compound device.</p>
9	NPS	<p>0</p> <p>1</p>	<p>No power switching</p> <p>Indicates that VBUS power switching is supported and is either per-port or all-port switched per the power switching mode field.</p> <p>Indicates that VBUS power switching is not supported and that power is available to all downstream ports when the USB host controller is powered. This is the default value.</p> <p>The RM4x microcontroller USB Host ports support over-current protection inputs and automatic power switching upon indication of an over-current condition. The USB Host ports must not be in the Reset state for the over-current protection inputs to take effect.</p>
8	PSM	<p>0</p> <p>1</p>	<p>Power switching mode</p> <p>Indicates that all ports are powered at the same time. This is the default value.</p> <p>Indicates that each port is powered individually. This mode allows port power to be controlled by either the global switch or per-port switching.</p> <p>If the PortPowerControlMask (PPCM) bit in the <a href="#">Section 28.2.4.20</a> register is set, the USB host port only responds to port power commands (Set/ClearPortPower). If the port mask is cleared, then the USB host port is controlled by the global power switch (Set/ClearGlobalPower).</p>

**Table 28-21. HC Root Hub A Register (HCRHDESCRIPTORA) Field Descriptions (continued)**

Bit	Field	Value	Description
7 - 0	NDP	2	<p>Number of downstream ports</p> <p>This register defaults to 2 to indicate two downstream ports.</p> <p>The USB signal multiplexing mode and top-level pin multiplexing features can place the device in a mode where 0, 1, or 2 of the USB host controller downstream ports are usable. This register reports two ports, regardless of top-level pin multiplexing mode.</p>

**28.2.4.20 HC Root Hub B Register (HCRHDESCRIPTORB)**

The HC root hub B register defines several aspects of the USB host controller root hub functionality.

**Figure 28-20. HC Root Hub B Register (HCRHDESCRIPTORB) [address = FCF78B4Ch]**


LEGEND: R = Read, W = Write, n = value at reset

**Table 28-22. HC Root Hub B Register (HCRHDESCRIPTORB) Bit Field Descriptions**

Bit	Field	Value	Description
31 - 16	PPCM		<p>Port power control mask</p> <p>Each bit defines whether a corresponding downstream port has port power controlled by the global power control. If set, per-port power control is implemented for the corresponding port. If clear, global power control is implemented for the corresponding port.</p> <p>PPCM bit 0 is reserved.</p> <p>PPCM bit 1 is the port power control mask for downstream port 0.</p> <p>PPCM bit 2 is the port power control mask for downstream port 1.</p> <p>PPCM bit 3 is the port power control mask for downstream port 2.</p> <p>PPCM bits 4 through 15 are reserved. The device does not provide connections from the USB host controller to pins to provide external port power switching.</p> <p>Systems that implement port power switching must use other mechanisms to control port power.</p> <p>System software can update these bits to simplify host controller driver coding.</p>
15 - 0	DR		<p>Device removable</p> <p>Each bit defines whether a corresponding downstream port has a removable or non-removable device. A cleared bit indicates the corresponding port may have a removable device attached. A set bit indicates that the corresponding port has a non-removable device attached.</p> <p>DR bit 0 is reserved..</p> <p>DR bit 1 is the device removable bit for downstream port 0.</p> <p>DR bit 2 is the device removable bit for downstream port 1.</p> <p>DR bit 3 is the device removable bit for downstream port 2.</p> <p>DR bits 4 through 15 are reserved.</p>

**28.2.4.21 HC Root Hub Status Register (HCRHSTATUS)**

The HC root hub status register reports the USB host controller root hub status.

**Figure 28-21. HC Root Hub Status Register (HCRHSTATUS) [address = FCF78B50h]**

31	30		18	17	16
CRWE		Reserved		OCIC	LPSC
R/W-0		R-0		R/W-0	R/W-0
15	14		2	1	0
DRWE		Reserved		OCI	LPS
R/W-0		R-0		R-0	R/W-0

LEGEND: R = Read, W = Write, n = value at reset

**Table 28-23. HC Root Hub Status Register (HCRHSTATUS) Bit Field Descriptions**

Bit	Field	Value	Description
31	CRWE		Clear remote wake-up enable Write of 0 has no effect. Write of 1 clears the device remote wake-up enable bit.
30 - 18	Reserved		Reserved
17	OCIC		Overcurrent indication change This bit is automatically set when the overcurrent indicator bit changes. Write of 0 has no effect. Write of 1 clears this bit. System software can update this register to simplify host controller driver coding.
16	LPSC		Local power status change This bit defaults to 0 because the root hub does not support the local power status feature. Write of 0 has no effect. Write of 1 sets the port power status bits for all ports if power switching mode is 0. A write of 1 sets port power status bits for ports with their corresponding port power control mask bits cleared if power switching mode is 1. System software can update this register to simplify host controller driver coding.
15	DRWE		Device remote wake-up enable When 1, this bit enables a connect status change event to be treated as a resume event, which causes a transition from USB suspend to USB resume state and sets the resume detected interrupt status bit. When 0, connect status change events do not cause a transition from USB suspend to USB resume state and the resume detected interrupt is not changed. Write of 0 has no effect. Write of 1 sets the device remote wake-up enable bit.
14 - 2	Reserved		Reserved
1	OCI		Overcurrent indicator This bit reports global overcurrent indication if global overcurrent reporting is selected. When 1, this bit indicates that an overcurrent condition has been sensed. When 0, no overcurrent condition has been sensed. Because the device does not provide signals for external hardware to report overcurrent status to the USB host controller, this bit is always 0.
0	LPS		Local power status The root hub does not support the local power status feature. This bit always reads as 0. Write of 0 has no effect. Write of 1 when in global power mode (power switching mode = 0), turns off power to all ports. If in per-port power mode (power switching mode = 1), a write of 1 turns off power to those ports whose corresponding port power control mask bit is 0. Because the device does not provide signals from the USB host controller to external VBUS switching circuitry, this bit has no effect. System software can update this register to simplify host controller driver coding.

**28.2.4.22 HC Port 0 Status and Control Register (HCRHPORTSTATUS0)**

The HC port 0 status and control register reports and controls the state of USB host port 0.

**Figure 28-22. HC Port 0 Status and Control Register (HCRHPORTSTATUS0) [address = FCF78B54h]**

31	Reserved						24
R-0							
23	21	20	19	18	17	16	
Reserved		PRSC	OCIC	PSSC	PESC	CSC	
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
15	Reserved				10	8	
R-0					LSDA/PPP	PPS/SPP	
					R/W-0	R/W-1	
7	5	4	3	2	1	0	
Reserved		PRS/SPR	POCI/CSS	PSS/SPS	PES/SPE	CCS/CPE	
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 28-24. HC Port 0 Status and Control Register (HCRHPORTSTATUS0) Field Descriptions**

Bit	Field	Value	Description
31 - 21	Reserved		Reserved
20	PRSC	0 1	Port 0 reset status change This bit indicates, when 1, that the port 0 port reset status bit has changed. Write of 0 has no effect. Write of 1 clears this bit.
19	OCIC		Port 0 overcurrent indicator change This bit indicates, when 1, that the port 0 port overcurrent indicator has changed. Write of 0 has no effect. Write of 1 clears this bit. The device does not provide inputs for signaling external overcurrent indication to the USB host controller. Overcurrent monitoring, if required, must be handled through some other mechanism.
18	PSSC	0 1	Port 0 suspend status change This bit indicates, when 1, that the port 0 port suspend status has changed. Suspend status is considered to have changed only after the resume pulse, low-speed EOP, and 3-ms synchronization delays have been completed. Write of 0 has no effect. Write of 1 clears this bit.
17	PESC	0 1	Port 0 enable status change This bit indicates, when 1, that the port 0 port enable status changed. Write of 0 has no effect. Write of 1 clears this bit.
16	CSC		Port 0 connect status change This bit indicates, when 1, that the port 0 current connect status has changed due to a connect or disconnect event. If current connect status is 0 when a set port reset, set port enable, or set port suspend write occurs, then this bit is set. A write of 1 clears this bit. A write of 0 has no effect. If the HCRHDESCRIPTORB.DR[1] bit is set to indicate a non-removable USB device on port 0, this bit is set only after a root hub reset to inform the system that the device is attached.
15 - 10	Reserved		Reserved

**Table 28-24. HC Port 0 Status and Control Register (HCRHPORTSTATUS0) Field Descriptions  
(continued)**

Bit	Field	Value	Description
9	LSDA/PPP		<p>Port 0 low-speed device attached/clear port power</p> <p>This bit indicates, when read as 1, that a low-speed device is attached to port 0. A 0 in this bit indicates a full-speed device.</p> <p>This bit is valid only when port 0 current connect status is 1.</p> <p>The host controller driver can write a 1 to this bit to clear the port 0 port power status. A write of 0 to this bit has no effect.</p> <p>The device USB host controller does not control external port power using OHCI mechanisms, so, if required, USB host port power must be controlled through other means.</p> <p>System software can update this register to simplify host controller driver coding.</p>
8	PPS/SPP		<p>Port 0 port power status/set port power</p> <p>This bit indicates, when read as 1, that the port 0 power is enabled. When read as 0, port 0 power is not enabled.</p> <p>The device does not provide signals from the USB host controller to control external port power, so if required, USB host port power control signals must be controlled through other means. Software can track the current power state using the port power status bit and other power control bits, but those bits have no direct effect on external port power control.</p> <p>System software can update this register to simplify host controller driver coding.</p> <p>A write of 1 to this bit sets the port 0 port power status bit. A write of 0 has no effect.</p> <p>System software can update this register to simplify host controller driver coding.</p>
7 - 5	Reserved		Reserved
4	PRS/SPR		<p>Port 0 port reset status/set port reset</p> <p>When read as 1, indicates that port 0 is signaling the USB reset. When read as 0, USB reset is not being sent to port 0.</p> <p>A write of 1 to this bit sets the port 0 port reset status bit and causes the USB host controller to begin signaling USB reset to port 0. A write of 0 to this bit has no effect.</p>
3	POCI/CSS		<p>Port 0 port overcurrent indicator/clear suspend status</p> <p>When read as 1, indicates a port 0 port overcurrent condition has occurred. When 0, no port 0 port overcurrent condition has occurred.</p> <p>The device does not provide inputs for signaling external overcurrent indication to the USB host controller. Overcurrent monitoring, if required, must be handled through some other mechanism.</p> <p>A write of 1 to this bit when port 0 port suspend status is 1 causes resume signaling on port 0. A write of 1 when port 0 port suspend status is 0 has no effect. A write of 0 has no effect.</p>
2	PSS/SPS		<p>Port 0 port suspend status/set port suspend</p> <p>When read as 1, indicates that port 0 is in the USB suspend state or is in the resume sequence. When 0, indicates that port 0 is not in the USB suspend state. This bit is cleared automatically at the end of the USB resume sequence and also at the end of the USB reset sequence.</p> <p>If port 0 current connect status is 1, a write of 1 to this bit sets the port 0 port suspend status bit and places port 0 in USB suspend state. If current connect status is 0, a write of 1 instead sets connect status change to inform the USB host controller driver software of an attempt to suspend a disconnected device. A write of 0 to this bit has no effect.</p>
1	PES/SPE		<p>Port 0 port enable status/set port enable</p> <p>When read as 1, indicates that port 0 is enabled. When read as 0, this bit indicates that port 0 is not enabled. This bit is automatically set at completion of port 0 USB reset if it was not already set before the USB reset completed, and is automatically set at the end of a USB suspend if the port was not enabled when the USB resume completed.</p> <p>A write of 1 to this bit when port 0 current connect status is 1 sets the port 0 port enable status bit. A write of 1 when port 0 current connect status is 0 has no effect. A write of 0 has no effect.</p>
0	CCS/CPE		<p>Port 0 current connection status/clear port enable</p> <p>When read as 1, indicates that port 0 currently has a USB device attached. When 0, indicates that no USB device is attached to port 0.</p> <p>This bit is set to 1 after root hub reset if the HCRHDESCRIPTORB.DR[1] bit is set to indicate a non-removable device on port 0.</p> <p>A write of 1 to this bit clears the port 0 port enable bit. A write of 0 to this bit has no effect.</p>

**28.2.4.23 HC Port 1 Status and Control Register (HCRHPORTSTATUS1)**

The HC port 1 status register reports and controls the state of USB host port 1.

**Figure 28-23. HC Port 1 Status and Control Register (HCRHPORTSTATUS1) [address = FCF78B58h]**

31	Reserved						24
R-0							
23	21	20	19	18	17	16	
Reserved		PRSC	OCIC	PSSC	PESC	CSC	
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
15	Reserved				10	9	8
R-0					LSDA/CPP	PPS/SPP	
					R/W-0	R/W-1	
7	5	4	3	2	1	0	
Reserved		PRS/SPR	POCI/CSS	PSS/SPS	PES/SPE	CCS/CPE	
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 28-25. HC Port 1 Status and Control Register (HCRHPORTSTATUS1) Field Descriptions**

Bit	Field	Value	Description
31 - 21	Reserved		Reserved
20	PRSC		Port 1 reset status change This bit indicates, when 1, that the port 1 port reset status bit has changed. A write of 1 clears this bit. A write of 0 has no effect.
19	OCIC		Port 1 overcurrent indicator change This bit indicates, when 1, that the port 1 port overcurrent indicator has changed. A write of 1 clears this bit. A write of 0 has no effect. The device does not provide inputs for signaling external overcurrent indication to the USB host controller. Overcurrent monitoring, if required, must be handled through some other mechanism.
18	PSSC		Port 1 suspend status changed This bit indicates, when 1, that the port 1 port suspend status has changed. Suspend status is considered to have changed only after the resume pulse, low-speed EOP, and 3-ms synchronization delays have been completed. A write of 1 clears this bit. A write of 0 has no effect.
17	PESC		Port 1 enable status change This bit indicates, when 1, that the port 1 port enable status changed. A write of 1 clears this bit. A write of 0 has no effect.
16	CSC		Port 1 connect status change This bit indicates, when 1, that the port 1 current connect status has changed due to a connect or disconnect event. If current connect status is 0 when a set port reset, set port enable, or set port suspend write occurs, then this bit is set. A write of 1 clears this bit. A write of 0 has no effect. If the HCRHDESCRIPTORB.DR[2] bit is set to indicate a nonremovable USB device on port 1, this bit is set only after a root hub reset to inform the system that the device is attached.
15 - 10	Reserved		Reserved



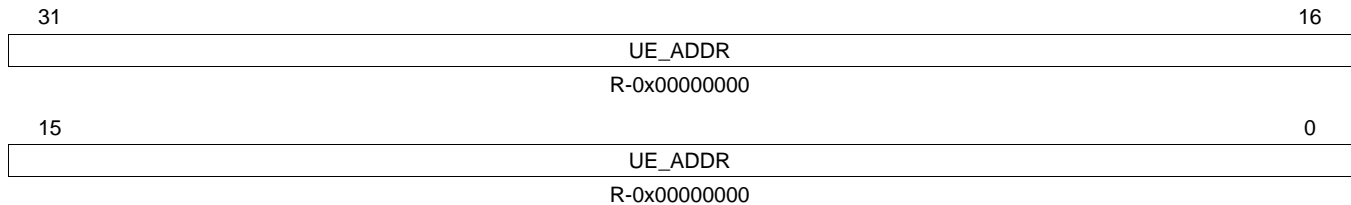
**Table 28-25. HC Port 1 Status and Control Register (HCRHPORTSTATUS1) Field Descriptions  
(continued)**

Bit	Field	Value	Description
9	LSDA/PPP		<p>Port 1 low-speed device attached/clear port power</p> <p>This bit indicates, when read as 1, that a low-speed device is attached to port 1. A 0 in this bit indicates a full-speed device.</p> <p>This bit is valid only when port 1 current connect status is 1.</p> <p>The host controller driver can write a 1 to this bit to clear the port 1 port power status. A write of 0 to this bit has no effect.</p> <p>The device USB host controller does not control external port power using OHCI mechanisms, so, if required, USB host port power must be controlled through other means.</p>
8	PPS/SPP		<p>Port 1 port power status/set port power</p> <p>This bit indicates, when read as 1, that the port 1 power is enabled. When read as 0, port 1 power is not enabled.</p> <p>The device does not provide signals from the USB host controller to control external port power, so, if required, USB host port power control signals must be controlled through other means. Software can track the current power state using the port power status bit and other power control bits, but those bits have no direct effect on external port power control.</p> <p>A write of 1 to this bit sets the port 1 port power status bit. A write of 0 has no effect.</p> <p>System software can update this register to simplify host controller driver coding.</p>
7 - 5	Reserved		Reserved
4	PRS/SPR		<p>Port 1 port reset status/set port reset</p> <p>When read as 1, indicates that port 1 is sending a USB reset. When read as 0, USB reset is not being sent to port 1.</p> <p>A write of 1 to this bit sets the port 1 port reset status bit and causes the USB host controller to begin signaling USB reset to port 1. A write of 0 to this bit has no effect.</p>
3	POCI/CSS		<p>Port 1 port overcurrent indicator/clear suspend status</p> <p>When read as 1, indicates that a port 1 port overcurrent condition has occurred. When 0, no port 1 port overcurrent condition has occurred.</p> <p>The device does not provide inputs for signaling external overcurrent indication to the USB host controller. Overcurrent monitoring, if required, must be handled through some other mechanism.</p> <p>A write of 1 to this bit when port 1 port suspend status is 1 causes resume signaling on port 1. A write of 1 when port 1 port suspend status is 0 has no effect. A write of 0 has no effect.</p>
2	PSS/SPS		<p>Port 1 port suspend status/set port suspend</p> <p>When read as 1, indicates that port 1 is in the USB suspend state, or is in the resume sequence. When 0, indicates that port 1 is not in the USB suspend state. This bit is cleared automatically at the end of the USB resume sequence and also at the end of the USB reset sequence.</p> <p>If port 1 current connect status is 1, a write of 1 to this bit sets the port 1 port suspend status bit and places port 1 in USB suspend state. If current connect status is 0, a write of 1 instead sets connect status change to inform the USB host controller driver software of an attempt to suspend a disconnected device. A write of 0 to this bit has no effect.</p>
1	PES/SPE		<p>Port 1 port enable status/set port enable</p> <p>When read as 1, indicates that port 1 is enabled. When read as 0, this bit indicates that port 1 is not enabled. This bit is automatically set at completion of port 1 USB reset if it was not already set before the USB reset completed and is automatically set at the end of a USB suspend if the port was not enabled when the USB resume completed.</p> <p>A write of 1 to this bit when port 1 current connect status is 1 sets the port 1 port enable status bit. A write of 1 when port 1 current connect status is 0 has no effect. A write of 0 has no effect.</p>
0	CCS/CPE		<p>Port 1 current connection status/clear port enable</p> <p>When read as 1, indicates that port 1 currently has a USB device attached. When 0, indicates that no USB device is attached to port 1.</p> <p>This bit is set to 1 after root hub reset if the HCRHDESCRIPTORB.DR[2] bit is set to indicate a non-removable device on port 1.</p> <p>A write of 1 to this bit clears the port 1 port enable bit. A write of 0 to this bit has no effect.</p>

### 28.2.4.24 Host UE Address Register (HOSTUEADDR)

The host UE address register reports the physical address of the last OCPI bus access that caused an unrecoverable error (UE). This register has no meaning until an unrecoverable error has occurred. It also has no meaning if the USB host controller issues an unrecoverable error because the offset checking fault occurred while processing an isochronous TD. This register is not defined by the OHCI specification.

**Figure 28-24. Host UE Address Register (HOSTUEADDR) [address = FCF78BE0h]**



LEGEND: R = Read, W = Write, n = value at reset

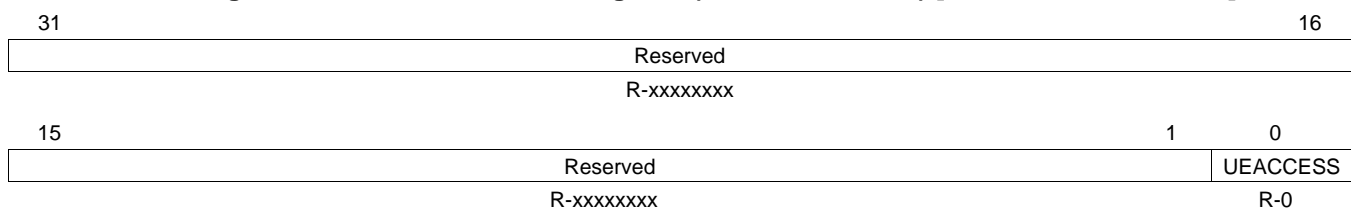
**Table 28-26. Host UE Address Register (HOSTUEADDR) Field Descriptions**

Bit	Field	Value	Description
31 - 0	UE_ADDR		Unrecoverable error address  This register captures the physical address of any bus transaction that is started by the USB host controller that encounters an unrecoverable error condition. This information, along with the information in HOSTUESTATUS, can help a developer determine why the USB host issued an access to a physical address that resulted in an unrecoverable error.

### 28.2.4.25 Host UE Status Register (HOSTUESTATUS)

The host UE status register reports the bus cycle-type for the last unrecoverable error that occurred. This register has no meaning until an unrecoverable error has occurred. It also has no meaning if the USB host controller issues an unrecoverable error because the offset checking fault occurred while processing an isochronous TD. This register is not defined by the OHCI specification.

**Figure 28-25. Host UE Status Register (HOSTUESTATUS) [address = FCF78BE4h]**



LEGEND: R = Read, W = Write, n = value at reset

**Table 28-27. Host UE Status Register (HOSTUESTATUS) Bit Field Descriptions**

Bit	Field	Value	Description
31 - 1	Reserved		Reserved
0	UEACCESS		Access type when unrecoverable error occurred  When an unrecoverable error occurs because of time-out of a bus write, this bit is set. When an unrecoverable error occurs because of time-out of a bus read, this bit is cleared. This bit has no meaning before an unrecoverable error occurs.  This information, along with the information in HOSTUEADDR, can help a developer determine why the USB host issued a bus access that resulted in an unrecoverable error.

### 28.2.4.26 Host Time-out Control Register (HOSTTIMEOUTCTRL)

The host time-out control register controls the USB host controller bus time-out mechanism. This register is not defined by the OHCI specification.

**Figure 28-26. Host Time-out Control Register (HOSTTIMEOUTCTRL) [address = FCF78BE8h]**

31	Reserved	16
R-0		
15	Reserved	TO_DIS
R-0		R/W-0

LEGEND: R = Read, W = Write, n = value at reset

**Table 28-28. Host Time-out Control Register (HOSTTIMEOUTCTRL) Bit Field Descriptions**

Bit	Field	Value	Description
31 - 1	Reserved		Reserved
0	TO_DIS		Bus time-out disable When 1, the USB host controller bus time-out counter is disabled and the host controller waits indefinitely for completion of a USB host controller access to system memory. When cleared (the default state), the USB host controller waits no more than 4096 VCLK3 clocks for completion of a bus access to system memory. If the bus cycle does not complete in that time, the USB host controller signals an unrecoverable error.

**28.2.4.27 Host Revision Register (HOSTREVISION)**

The host revision register returns the revision number for the USB host controller. This register is not defined by the OHCI specification.

**Figure 28-27. Host Revision Register (HOSTREVISION) [address = FCF78BECh]**

31	Reserved	16
R-xxxxxx		
15	Reserved	MAJORREV
8	7	MINORREV
R-xxxxxx		R-xx

LEGEND: R = Read, W = Write, n = value at reset

**Table 28-29. Host Revision Register (HOSTREVISION) Field Descriptions**

Bit	Field	Value	Description
31 - 8	Reserved		Reserved
7 - 4	MAJORREV		Major revision number MAJORREV indicates the major revision number of the USB host controller. The original USB host controller version implements a major revision number of 0.
3 - 0	MINORREV		Minor revision number MINORREV indicates the minor revision number of the USB host controller. The original USB host controller version implements a minor revision number of 0.

**28.2.5 USB Host Controller Reserved Registers and Reserved Bit Fields**

To enhance code reusability with possible future versions of the USB host controller, reads and writes to reserved USB host controller register addresses are to be avoided. Unless otherwise specified, when writing registers that have reserved bits, read-modify-write operations must be used so that the reserved bits are written with their previous values.

## **28.2.6 USB Host Controller Registers, USB Reset, and USB Clocking**

The Global Clock Module (GCM) provides VCLK3 to the USB host controller for the bus accesses. The GCM also provides the 48MHz clock (VCLKA3\_S) and 12MHz clock (VCLKA3\_DIVR) to the USB host controller. If the application disables the VCLK3 domain or the VCLKA3 domain from the GCM, reads from and writes to the USB host controller registers do not occur correctly. To properly access the USB host controller registers, the USB host controller must be clocked and must be out of reset.

The USB host controller hardware reset is controlled by the system reset signal from the SYS module. The USB host controller is held in reset whenever the system reset signal is low or the HostControllerReset (HCR) field of the HcCommandStatus Register is 1. The USB host controller can transition out of reset whenever the clock is enabled and system reset is high and HcCommandStatus.HCR is 0.

The USB host controller completes its reset within about 72 cycles of the 48-MHz clock after the host controller clock transitions out of reset. After system software turns on the clock to the USB host controller and removes it from reset, it is necessary to wait until the USB host controller internal reset completes. To ensure that the USB host controller has completely reset, system software must wait until reads of both the HCREVISION register and the HCHCCA register return their correct reset default values.

## **28.2.7 OHCI Interrupts**

The USB host controller provides an interrupt output to the Vectored Interrupt Manager (VIM). This is a level-sensitive interrupt signal.

### **28.2.7.1 OHCI Scheduling Overrun Interrupt**

The OHCI scheduling overrun interrupt is supported as described in the *OHCI Specification for USB*.

### **28.2.7.2 OHCI HCDONEHEAD Writeback Interrupt**

The OHCI HCDONEHEAD writeback interrupt is supported as described in the *OHCI Specification for USB*.

### **28.2.7.3 OHCI Start Of Frame Interrupt**

The OHCI start of frame interrupt is supported as described in the *OHCI Specification for USB*.

### **28.2.7.4 OHCI Resume Detect Interrupt**

The OHCI resume detect interrupt is supported as described in the *OHCI Specification for USB*.

### **28.2.7.5 OHCI Unrecoverable Error Interrupt**

The OHCI unrecoverable error interrupt is supported as described in the *OHCI Specification for USB*. This interrupt occurs if the USB host controller is unable to complete a bus read or write within 4096 VCLK3 clocks when the USB host bus time-out feature is enabled. When a bus time-out causes an unrecoverable error, HOSTUEADDR and HOSTUESTATUS are updated. When an isochronous TD is processed with an OFFSET/PSW field that is not set for *Not Accessed*, an unrecoverable error interrupt is generated, but HOSTUEADDR and HOSTUESTATUS are not updated.

### **28.2.7.6 OHCI Frame Number Overflow**

The OHCI frame number overflow interrupt is supported as described in the *OHCI Specification for USB*.

### **28.2.7.7 OHCI Root Hub Status Change**

The OHCI root hub status change interrupt is supported as described in the *OHCI Specification for USB*. The device does not provide a connection between the USB host controller and USB port overcurrent detection hardware, so the root hub status change interrupt does not occur because of a port overcurrent event.

### 28.2.7.8 OHCI Ownership Change Interrupt

The optional OHCI ownership change interrupt is not supported.

### 28.2.8 USB Host Controller Access to System Memory

The USB host controller has access to system memory to read and write the OHCI data structures and data buffers associated with USB traffic.

### 28.2.9 Physical Addressing

All system memory accesses initiated by the USB host controller use physical addresses.

### 28.2.10 USB Host Controller Bus Addressing and OHCI Data Structure Pointers

The USB host controller OHCI registers that point to the HCCA and the ED lists must be programmed with values that correspond to the physical addresses of the particular data structure. System software must use physical addresses when manipulating the OHCI control registers that point to the HCCA, and to the ED and TD lists. System software must also use physical addresses for the ED and TD pointers that are stored within ED and TD entries.

The USB host controller driver software must also be able to examine the list of completed transfer descriptors that the host controller creates as it retires transfer descriptors. This list is pointed to by the HCDONEHEAD register, which contains a physical address that points to the most recent transfer descriptor that has been retired.

### 28.2.11 NULL Pointers

The *OHCI Specification for USB* uses NULL pointers to indicate the end of a list. The USB host controller compares the ED and TD pointers against the value 0x00000000 to determine if the pointer is a null pointer. Address conversion routines must understand this usage.

### 28.2.12 USB Host Controller Power Management

Power management of the USB host controller is limited to disabling the clock to the USB host by clearing UHOST\_EN. When UHOST\_EN is 0, the USB host controller clocks are disabled and the USB host controller is held in reset. The USB signal multiplexing controlled by register in the I/O Multiplexing Module (IOMM).

When the host controller 48-MHz clock is disabled or UHOST\_EN is 0, all USB host controller OHCI registers and the HOSTUEADDR, HOSTUESTATUS, HOSTTIMEOUTCTRL, and HOSTREVISION registers are inaccessible.

## 28.3 USB Device Controller

The USB device controller supports the implementation of a full-speed device fully compliant with the USB 1.1 standard.

It provides an interface between the host processor and the USB wire and handles USB transactions with minimal CPU software intervention.

The USB device controller module supports one control endpoint (EP0), up to 15 IN endpoints, and up to 15 OUT endpoints. The exact endpoint configuration is software programmable. The specific items of a configuration are for each endpoint, the size in bytes, the direction (IN, OUT), the type (bulk/interrupt or ISO), and the associated number.

The USB device controller module also supports three DMA channels for IN endpoints and three DMA channels for OUT endpoints for either bulk/interrupt or ISO transactions.

### 28.3.1 USB Device Controller Registers

[Table 28-30](#) lists the USB device controller registers. These registers are aligned at a 16-bit boundary, so that the memory address offsets for the registers are 0x0, 0x2, 0x4, 0x6 and so on. The registers support 8-bit and 16-bit read/write accesses. A 32-bit read access will return the same contents on the upper 16 bits as on the lower 16 bits. [Section 28.3.1.1](#) through [Section 28.3.1.23](#) describe the register bits.

**Table 28-30. USB Device Controller Registers**

Register	Description	Address	Section
REV	Revision	FCF78A00h	<a href="#">Section 28.3.1.1</a>
<b>Endpoint</b>			
EP_NUM	Endpoint selection	FCF78A02h	<a href="#">Section 28.3.1.2</a>
DATA	Data	FCF78A04h	<a href="#">Section 28.3.1.3</a>
CTRL	Control	FCF78A06h	<a href="#">Section 28.3.1.4</a>
STAT_FLG	Status	FCF78A08h	<a href="#">Section 28.3.1.5</a>
RXFSTAT	Receive FIFO status	FCF78A0Ah	<a href="#">Section 28.3.1.6</a>
SYSCON1	System configuration 1	FCF78A0Ch	<a href="#">Section 28.3.1.7</a>
SYSCON2	System configuration 2	FCF78A0Eh	<a href="#">Section 28.3.1.8</a>
DEVSTAT	Device status	FCF78A10h	<a href="#">Section 28.3.1.9</a>
SOF	Start of frame	FCF78A12h	<a href="#">Section 28.3.1.10</a>
IRQ_EN	Interrupt enable	FCF78A14h	<a href="#">Section 28.3.1.11</a>
DMA_IRQ_EN	DMA interrupt enable	FCF78A16h	<a href="#">Section 28.3.1.12</a>
IRQ_SRC	Interrupt source	FCF78A18h	<a href="#">Section 28.3.1.13</a>
EPN_STAT	Non-ISO endpoint interrupt enable	FCF78A1Ah	<a href="#">Section 28.3.1.14</a>
DMAN_STAT	Non-ISO DMA interrupt enable	FCF78A1Ch	<a href="#">Section 28.3.1.15</a>
<b>DMA Configuration</b>			
RXDMA_CFG	DMA receive channels configuration	FCF78A20h	<a href="#">Section 28.3.1.16</a>
TXDMA_CFG	DMA transmit channels configuration	FCF78A22h	<a href="#">Section 28.3.1.17</a>
DATA_DMA	DMA FIFO data	FCF78A24h	<a href="#">Section 28.3.1.18</a>
TXDMA0	Transmit DMA control 0	FCF78A26h	<a href="#">Figure 28-44</a>
TXDMA1	Transmit DMA control 1	FCF78A28h	<a href="#">Figure 28-44</a>
TXDMA2	Transmit DMA control 2	FCF78A2Ah	<a href="#">Figure 28-44</a>
RXDMA0	Receive DMA control 0	FCF78A30h	<a href="#">Figure 28-43</a>
RXDMA1	Receive DMA control 1	FCF78A32h	<a href="#">Figure 28-43</a>
RXDMA2	Receive DMA control 2	FCF78A34h	<a href="#">Figure 28-43</a>
<b>Endpoint Configuration</b>			
EP0	Endpoint configuration 0	FCF78A40h	<a href="#">Section 28.3.1.21</a>
EP1_RX...EP15RX	Receive endpoint configuration 1...15	FCF78A42h	<a href="#">Section 28.3.1.22</a>
		FCF78A5Eh	

**Table 28-30. USB Device Controller Registers (continued)**

Register	Description	Address	Section
EP1_TX...EP15TX	Transmit endpoint configuration 1...15	FCF78A62h — FCF78A7Eh	<a href="#">Section 28.3.1.22</a>

**28.3.1.1 Revision Register (REV)**

This read-only register contains the revision number of the module. A write to this register is denied. Device reset and USB controller reset have no effect on this register.

**Figure 28-28. Revision Register (REV) [address = FCF78A00h]**

15	Reserved	8
	R-0x00	
7	REV_NB	0
	R-0x61	

LEGEND: R = Read, W = Write, n = value at reset

**Table 28-31. Revision Register (REV) Bit Field Descriptions**

Bit	Field	Value	Description
15 - 8	Reserved		Reserved. Reads return zeros, writes have no effect.
7 - 0	REV_NB	0x61	This 8-bit field indicates revision number of current USB device controller module—value fixed by hardware: Revision 6.1

**28.3.1.2 Endpoint Selection Register (EP\_NUM)**

This read/write register selects and enables the endpoint that can be accessed by the USB device controller.

**Figure 28-29. Endpoint Selection Register (EP\_NUM) [address = FCF78A02h]**

15	Reserved				8
	R-0x00				
7	6	5	4	3	0
Reserved	SETUP_SEL	EP_SEL	EP_DIR	EP_NUM	
R-0	RW-0	RW-0	RW-0	RW-0x0	

LEGEND: R = Read, W = Write, n = value at reset

**Table 28-32. Endpoint Selection Register (EP\_NUM) Bit Field Descriptions**

Bit	Field	Value	Description
15 - 7	Reserved		Reserved. Reads return zeros, writes have no effect.
6	SETUP_SEL	0 1	Set by the CPU in response to a Setup general USB Interrupt, to access the EP0 read-only Setup FIFO when reading the DATA register. Setting this bit will clear the IRQ_SRC.Setup bit. When this bit is set, other EP_NUM register bits must be 0. Note: After having read the Setup FIFO, the CPU must clear this bit by writing 0 to it. 0 No access 1 Access permitted The value after system reset or USB reset is low.



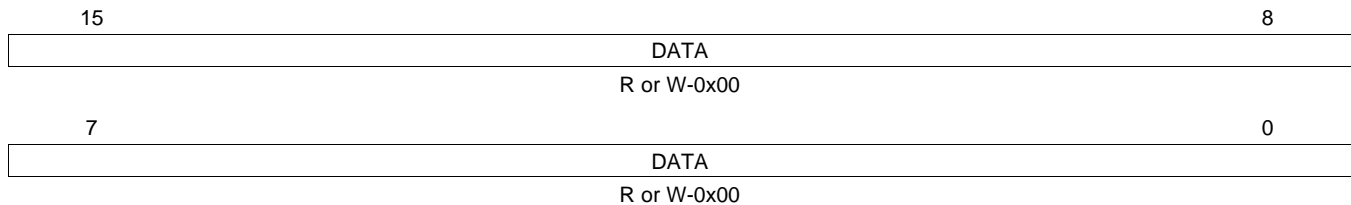
**Table 28-32. Endpoint Selection Register (EP\_NUM) Bit Field Descriptions (continued)**

Bit	Field	Value	Description
5	EP_SEL	0 1	Set by the CPU to access the status (STAT_FLG, RXFSTAT) and data (DATA) registers for the endpoint selected. If the EP_NUM.EP_DIR bit is set to 0, the CPU can read data from endpoint RX FIFO by reading the DATA register. If the EP_NUM.EP_DIR bit is set to 1, the CPU can write data into endpoint TX FIFO by writing into the DATA register. After each access to an endpoint during interrupt handling, the CPU must absolutely clear this bit. Note: When the CPU sets this bit, it must set the SETUP_SEL bit to 0. After having accessed the endpoint's FIFO for either read or write access, the CPU must clear this bit by writing a 0 to it. 0 No access 1 Access permitted. Value after system reset or USB reset is low.
4	EP_DIR	0 1	The endpoint direction bit gives the direction associated with the endpoint number selected in EP_NUM.EP_NUM: 0 OUT endpoint 1 IN endpoint Value after system reset or USB reset is low.
3 - 0	EP_NUM	0000 0001 ... 1111	The endpoint number binary encoded in these four bits, associated with the direction given by EP_NUM.EP_DIR bit, is the current endpoint selected. All reads and writes to the endpoint status and the control and data locations are for this endpoint. EP0 EP1 ... EP15 Value after system reset or USB reset is low.

**28.3.1.3 Data Register (DATA)**

This register is the entry point to write into a selected TX endpoint or to read data from a selected RX endpoint, or to read data from setup FIFO. If selected endpoint direction is OUT, this register is read-only and a write into it is denied. If selected endpoint direction is IN, this register is write-only and a read to this register is denied.

**Figure 28-30. Data Register (DATA) [address = FCF78A04h]**



LEGEND: R = Read, W = Write, n = value at reset

**Table 28-33. Data Register (DATA) Bit Field Descriptions**

Bit	Field	Value	Description
15 - 0	DATA		Transmit/receive FIFO data: EP_NUM.EP_DIR = 1: This register contains the data written by the USB device controller to be sent to the USB host during the next IN transaction. Data can be written successfully only if the EP_NUM.EP_SEL bit is asserted. EP_NUM.EP_DIR = 0: This register contains the data received by the USB core from USB host OUT or SETUP transactions. Data can be read successfully only if the EP_NUM.EP_SEL bit is asserted, or if EP_NUM.SETUP_SEL bit is asserted (for setup data).



### 28.3.1.4 Control Register (CTRL)

This set-only register controls the FIFO and status of the selected endpoint. A read access to this register always returns 0.

Endpoint 0 setup FIFO is always enabled and ready to accept setup data. No control register CTRL is implemented for this FIFO because the USB device controller cannot control it.

**Figure 28-31. Control Register (CTRL) [address = FCF78A06h]**

Reserved							8
R-0x00							
7	6	5	3	2	1	0	
CLR_HALT	SET_HALT	Reserved		SET_FIFO_EN	CLR_EP	RESET_EP	
RW-0	RW-0	R-000		RW-0	RW-0	RW-0	

LEGEND: R = Read, W = Write, n = value at reset

**Table 28-34. Control Register (CTRL) Bit Field Descriptions**

Bit	Field	Value	Description
15 - 8	Reserved		Reserved
7	CLR_HALT	0 1	The clear halt endpoint (non-ISO) bit only concerns non-ISO endpoints—used by the USB device controller to clear an endpoint halt condition:  No action Clear halt condition  Always read 0
6	SET_HALT	0 1	The set halt endpoint (non-ISO) only concerns non-ISO endpoints—used by the USB device controller to halt the selected endpoint.  The halted endpoint returns STALL handshakes to the USB host. The USB device controller can disable the endpoint interrupt if it does not want to be informed of STALL handshakes.  Note: If the endpoint to halt is used by a DMA channel, the USB device controller must disable the DMA channel before setting halt condition for this endpoint.  No action Halt endpoint  Always read 0
5 - 3	Reserved		Reserved. Reads return zeros, writes have no effect.
2	SET_FIFO_EN	0 1	The set FIFO enable (non-ISO) bit only concerns non-ISO endpoints. If the selected endpoint direction is IN, the USB device controller uses this bit to enable the USB device to transmit data from the FIFO at the next valid IN token. If the selected endpoint direction is OUT, the USB device controller uses this bit to enable the USB device to receive data from the USB host at the next valid OUT transaction. If not, setting the device returns a NAK handshake. ISO endpoints FIFO are always enabled.  Note: The USB device controller must never enable endpoint 0 FIFO out of control transfers. For bulk and interrupt endpoints, FIFO must never be enabled when the halt feature is set or when RX FIFO is not empty. Furthermore, during EP interrupts handling, the USB device controller must have cleared the interrupt bit before setting CTRL.SET_FIFO_EN bit (to avoid masked ACK interrupts).  No action FIFO enabled  Always read 0
1	CLR_EP	0 1	Clear endpoint: the USB device controller sets this bit to clear the selected endpoint FIFO pointers and flags. This bit resets the FIFO pointers, the FIFO empty status bit is set, and the FIFO enable bit and other FIFO flags are cleared upon completion of the FIFO reset. It also clears the previous transaction handshake status. For ISO endpoints or non-ISO double-buffered endpoints, both foreground and background FIFO are cleared.  No action Clear endpoint  Always read 0

**Table 28-34. Control Register (CTRL) Bit Field Descriptions (continued)**

Bit	Field	Value	Description
0	RESET_EP		The endpoint reset (non-control) bit only concerns non-control endpoints. The USB device controller sets this bit to reset the selected endpoint. It forces an interrupt or a bulk endpoint data PID to DATA0, clears the halt condition, and clears the FIFO (both foreground and background if endpoint is double-buffered) and previous transactions handshake status. For an ISO endpoint, it only clears the FIFO (both foreground and background).
		0	No action
		1	Reset endpoint
			Always read 0

### 28.3.1.5 Status Register (STAT\_FLG)

This read-only register provides a status of the FIFO and the results of the transactions handshakes for the selected endpoint. The 8 MSB are reserved for ISO endpoints, whereas the 8 LSB are reserved for non-ISO endpoints. This register cannot be read if EP\_NUM.EP\_SEL bit is not asserted for the endpoint. No status flag exists for the read-only set-up FIFO, which is always enabled.

**NOTE:** The updates for non-ISO transactions are done at the end of each non-transparent and valid transaction to a given endpoint, if no non-handled interrupt is pending on the endpoint.

The definition of a non-transparent, non-ISO IN transaction is one that responds with an ACK handshake or a STALL handshake, or optionally a NAK handshake if SYSCON1.NAK\_EN is asserted to 1. An ERR handshake or a NAK handshake when SYSCON1.NAK\_EN is 0 is considered transparent.

A write to this register has no effect.

**Figure 28-32. Status Register (STAT\_FLG) [address = FCF78A08h]**

15	14	13	12	11	10	9	8
NO_RX_PACKET	MISS_IN	DATA_FLUSH	ISO_ERR	Reserved		ISO_FIFO_EMPTY	ISO_FIFO_FULL
R-0	R-0	R-0	R-0	R-00		R-0	R-0
7	6	5	4	3	2	1	0
Reserved	EP_HALTED	STALL	NAK	ACK	FIFO_EN	NON_ISO_FIFO_EMPTY	NON_ISO_FIFO_FULL
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read, W = Write, n = value at reset

**Table 28-35. Status Register (STAT\_FLG) Bit Field Descriptions**

Bit	Field	Value	Description
15	NO_RX_PACKET		The isochronous no packet received (ISO OUT) bit only concerns ISO OUT endpoints. This bit notifies the USB device controller that the core has not received any isochronous packet on the endpoint. This bit is updated on a SOF (start of frame).
		0	The endpoint received a packet on the previous frame.
		1	The endpoint did not receive a packet on the previous frame.
			Value after system reset or USB reset is high.
14	MISS_IN		The isochronous missed IN token for the previous frame (ISO IN) bit only concerns ISO IN endpoints. This bit notifies the USB device controller that the core missed a valid ISO IN token during the previous frame and that TX data were flushed from the FIFO instead of being transmitted to the USB host.
		0	This bit is updated on a SOF (start of frame).
		1	The endpoint received an IN token the previous frame.
			The endpoint did not receive an IN token the previous frame and TX data were flushed.
			Value after system reset or USB reset is low.

**Table 28-35. Status Register (STAT\_FLG) Bit Field Descriptions (continued)**

Bit	Field	Value	Description
13	DATA_FLUSH	<p>0 Not significant</p> <p>1 Data was flushed.</p> <p>Value after system reset or USB reset is low.</p>	<p>The isochronous receive data flush (ISO OUT) bit only concerns ISO OUT endpoints. When set, this bit indicates that data were flushed from the isochronous FIFO that was moved from the foreground to the background. This happens when the USB device controller does not read all of the data from the foreground FIFO in a frame.</p> <p>This bit is updated in every frame.</p>
12	ISO_ERR	<p>0 Not significant</p> <p>1 ISO packet received with errors</p> <p>Value after system reset or USB reset is low.</p>	<p>The isochronous receive data error (ISO OUT) bit only concerns ISO OUT endpoints. When set, this bit indicates that the ISO data packet was received incorrectly. This happens when the core detects an error in the data packet (CRC, bit stuffing, PID check) or when there is an overrun condition in the FIFO. When this bit is set, the FIFO contents are automatically flushed by the core and the FIFO status is empty.</p> <p>This bit is updated in every frame.</p>
11 - 10	Reserved		Reserved
9	ISO_FIFO_EMPTY	<p>0 ISO FIFO is not empty.</p> <p>1 ISO FIFO is empty.</p> <p>Value after system reset or USB reset is high (FIFO empty).</p>	<p>The ISO FIFO empty bit only concerns ISO endpoints. This bit is set when the FIFO for the selected ISO endpoint is empty, either via an appropriate CTRL.CLR_EP bit or CTRL.RESET_EP bit or after successful reads from the selected FIFO.</p>
8	ISO_FIFO_FULL	<p>0 ISO FIFO is not full.</p> <p>1 ISO FIFO is full.</p> <p>Value after system reset or USB reset is low (FIFO empty).</p>	<p>The ISO FIFO full bit only concerns ISO endpoints. This bit is set when the FIFO for the selected ISO endpoint is full. This condition is cleared by setting the CTRL.CLR_EP bit or CTRL.RESET_EP bit, or after one successful read (by the USB device controller or the USB host).</p>
7	Reserved		Reserved
6	EP_HALTED	<p>0 The selected endpoint is not halted.</p> <p>1 The selected endpoint is halted.</p> <p>Value after system reset or USB reset is low.</p>	<p>The endpoint halted flag (non-ISO) bit only concerns non-ISO endpoints. This bit when asserted 1 indicates that the selected endpoint is halted. The endpoint can be put into the halt state only by the USB device controller writing the endpoint halt control bit (in response to a SET_FEATURE request, for instance).</p>
5	STALL	<p>0 No STALL handshake was returned.</p> <p>1 A STALL handshake packet was returned.</p> <p>Value after system reset or USB reset is low.</p>	<p>The transaction stall (non-ISO) bit only concerns non-ISO endpoints. This status bit is set at the end of a transaction if a STALL handshake packet was returned to the USB host, and if no non-handled interrupt is pending on the current buffer (see <a href="#">Section 28.3.11</a>). The core automatically returns a STALL packet if a valid IN token is received by a halted TX endpoint, if a valid OUT transaction is received by a halted RX endpoint, or if there is a request error (endpoint 0). The bit is cleared when the USB device controller has finished handling the corresponding interrupt (at EP_NUM.EP_SEL bit deselection).</p>

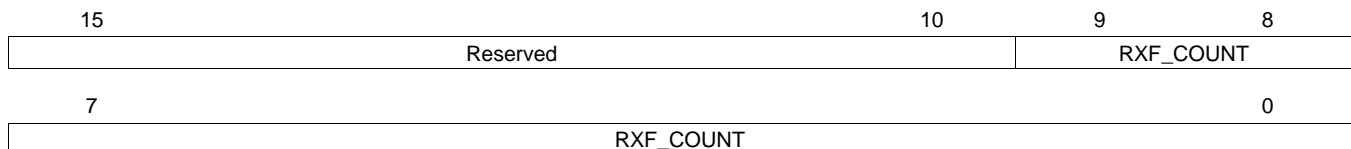
**Table 28-35. Status Register (STAT\_FLG) Bit Field Descriptions (continued)**

Bit	Field	Value	Description
4	NAK	0 1	The transaction non-acknowledge (non-ISO) bit only concerns non-ISO endpoints with SYSCON1.NAK_EN bit asserted. This status bit is set at the end of a transaction if a NAK handshake is returned to the USB host, and if no non-handled interrupt is pending on the current buffer. The USB core automatically returns a NAK handshake to the USB host if a valid IN token is received by a TX endpoint or if a valid OUT transaction is received by an RX endpoint, and the STAT_FLG.FIFO_EN bit is not set for the endpoint. The bit is cleared when the USB device controller has finished handling the corresponding interrupt (at EP_NUM.EP_SEL bit deselection).  0 No NAK handshake was returned (SYSCON1.NAK_EN bit is set). 1 A NAK handshake packet was returned and SYSCON1.NAK_EN bit is set. Value after system reset or USB reset is low.
3	ACK	0 1	The transaction acknowledge (non-ISO) bit only concerns non-ISO endpoints. This bit is set at the end of a non-transparent valid IN transaction if the data packet was sent successfully to the USB host and the ACK handshake was received, or at the end of a non-transparent valid OUT transaction if the data packet was received successfully by the USB device and the ACK handshake was returned. The bit is cleared when the USB device controller has finished handling the corresponding interrupt (at EP_NUM.EP_SEL bit deselection).  0 No ACK handshake packet was returned. 1 An ACK handshake packet was returned. Value after system reset or USB reset is low.
2	FIFO_EN	0 1	The FIFO enable status (non-ISO) bit only concerns non-ISO endpoints. This bit is asserted when CTRL.SET_FIFO_EN is set to 1 and is cleared automatically after a transaction completes with an ACK, STALL.  0 The non-ISO endpoint FIFO is disabled. 1 The non-ISO endpoint FIFO is enabled. Value after system reset or USB reset is low.
1	NON_ISO_FIFO_EMPTY	0 1	The non-ISO FIFO empty bit only concerns non-ISO endpoints. This bit is set when the FIFO for the selected non-ISO endpoint is empty, either via an appropriate CTRL.CLR_EP bit or CTRL.RESET_EP bit or after successful reads from the selected FIFO.  0 Non-ISO FIFO is not empty. 1 Non-ISO FIFO is empty. Value after system reset or USB reset is high (FIFO empty).
0	NON_ISO_FIFO_FULL	0 1	The non-ISO FIFO full bit only concerns non-ISO endpoints. This bit is set when the FIFO for the selected non-ISO endpoint is full. This condition is cleared by setting the CTRL.CLR_EP bit or CTRL.RESET_EP bit, or after one successful read (by the USB device controller or the USB host).  0 Non-ISO FIFO is not full. 1 Non-ISO FIFO is full. Value after system reset or USB reset is low (FIFO empty).

**28.3.1.6 Receive FIFO Status Register (RXFSTAT)**

This read-only register indicates how many bytes are in the receive FIFO for the selected endpoint. A write to this register has no effect. The USB device controller cannot read this register if the EP\_NUM.EP\_SEL bit is not set for the endpoint. No receive FIFO status exists for the setup FIFO, because 8 bytes are always expected.

**Figure 28-33. Receive FIFO Status Register (RXFSTAT) [address = FCF78A0Ah]**



LEGEND: R = Read, W = Write, n = value at reset

**Table 28-36. Receive FIFO Status Register (RXFSTAT) Bit Field Descriptions**

Bit	Field	Value	Description
15 - 10	Reserved		Reserved
9 - 0	RXF_COUNT		The receive FIFO byte count 10-bit field indicates the number of bytes currently in the receive FIFO. Values after system reset or USB reset is low (all 10 bits).

**28.3.1.7 System Configuration Register 1 (SYSCON1)**

This read/write register provides control functions for power management and miscellaneous control for the core.

Values depend on whether the reset action comes from the USB device controller (CPU) or the USB host.

**Figure 28-34. System Configuration Register 1 (SYSCON1) [address = FCF78A0Ch]**

15							9	8
Reserved							CFG_LOCK	
7	6	5	4	3	2	1	0	
DATA_ENDIAN	DMA_ENDIAN	Reserved	NAK_EN	AUTODEC_DIS	SELF_PWR	SOFF_DIS	PULLUP_EN	

LEGEND: R = Read, W = Write, n = value at reset

**Table 28-37. System Configuration Register 1 (SYSCON1) Bit Field Descriptions**

Bit	Field	Value	Description
15 - 9	Reserved		Reserved
8	CFG_LOCK	0 1	Device configuration locked bit: after the USB device controller has entered the device configuration (registers 0x20 to 0x3F), it must set the CFG_LOCK bit so that the device can be used. When the device configuration is not locked, the device is not ready to be used. Device configuration is not locked. Device is not ready Device configuration is locked. The value after the USB device controller hardware reset is low; after USB reset, it is unchanged (keep previous configuration).
7	DATA_ENDIAN	0 1	The data endian bit can be set by the USB device controller to select little- or big-endian format on data access (read or write). Little endian Big endian The value after USB device controller hardware reset is low; after USB reset, it is unchanged (keep previous configuration).
6	DMA_ENDIAN	0 1	The DMA data endian bit can be set by the USB device controller to select little- or big-endian format on data access (read or write). Little endian Big endian The value after USB device controller hardware reset is low; after USB reset, it is unchanged (keep previous configuration).
5	Reserved		Reserved
4	NAK_EN	0 1	The NAK enable bit can be set by the USB device controller to be signaled for NAK transaction handshake response. When this bit is set, STAT_FLG.NAK is set on a NAK handshake if no non-handled interrupt is pending on the endpoint, and the endpoint interrupt is asserted. In the normal mode, when cleared, NAK handshake response to the USB host is made transparent to the USB device controller and no interrupt is asserted. NAK is disabled. NAK is enabled. The value after system reset or USB reset is low. Note: If the USB device controller sets this bit, it must wait for a NAK interrupt before selecting TX endpoint to write TX data.

**Table 28-37. System Configuration Register 1 (SYSCON1) Bit Field Descriptions (continued)**

Bit	Field	Value	Description
3	AUTODEC_DIS	0 1	The autodecode process disabled can be set to force software to handle all EP0 transactions (except the SET_ADDRESS transaction). 0 Autodecode process is activated (see Table 28-55). 1 Autodecode process is deactivated. The value after USB device controller hardware reset is low; after USB reset, it is unchanged.
2	SELF_PWR	0 1	The self-powered bit indicates to the USB host whether the device is bus-powered or self-powered. This information is needed for an autodecoded request. The USB device controller must update this bit after a SET_CONFIGURATION according to the self-powered bit D6 given in the configuration descriptor (see USB 1.1 Specification). 0 Bus powered 1 Self-powered The value after USB device controller hardware reset is low; after USB reset, it is unchanged.
1	SOFF_DIS	0 1	When the shutoff disable bit is set, it disables the power shutoff circuitry. 0 Power shutoff circuitry is enabled. 1 Power shutoff circuitry is disabled. The value after USB device controller hardware reset is low; after USB reset, it is unchanged.
0	PULLUP_EN	0 1	The external pullup enable bit allows the device to disconnect itself from the USB bus, forcing the host to reset and reconfigure the device. This bit can be used to prevent USB traffic when the device is not ready. 0 Pull-up is disabled. USB host cannot detect the device. In this mode, the 48-MHz USB clock is forced off. 1 Pull-up is enabled. The value after USB device controller hardware reset is low; after USB reset, it is high; and after detach, it is unchanged.

**Table 28-38. Little Endian and Big Endian Formats**

<b>15 8</b>	<b>7 0</b>		<b>15 8</b>	<b>7 0</b>
Byte 1	Byte 0		Byte 0	Byte 1
<b>Little Endian Format (16 bits)</b>			<b>Big Endian Format (16 bits)</b>	

### 28.3.1.8 System Configuration Register 2 (SYSCON2)

This set-only register provides miscellaneous controls for the function. A read to this register always returns 0.

**Figure 28-35. System Configuration Register 2 (SYSCON2) [address = FCF78A0Eh]**

15	Reserved						8
7	6	5	4	3	2	1	0
Reserved	RMT_WKP	STALL_CMD	Reserved	DEV_CFG	CLR_CFG	Reserved	

LEGEND: R = Read, W = Write, n = value at reset

**Table 28-39. System Configuration Register 2 (SYSCON2) Bit Field Descriptions**

Bit	Field	Value	Description
15 - 7	Reserved		Reserved
6	RMT_WKP	0 1	The set-only remote wake-up bit, when written with a 1, initiates the remote wake-up sequence even if DEVSTAT.R_WK_OK bit was not previously set to 1 by the USB host. Reading this bit always returns 0. Writing 0 into this bit has no effect. To generate a resume, the software must check the remote wake-up enable value before initiating any wake-up sequence. 0 No action 1 Initiates the remote wake-up sequence Always read 0

**Table 28-39. System Configuration Register 2 (SYSCON2) Bit Field Descriptions (continued)**

Bit	Field	Value	Description
5	STALL_CMD	0 1	The set-only stall command bit only concerns non-autodecoded requests on control endpoint (EP0). This is asserted in response to a USB command where either the command itself or its data is invalid. Asserting this bit forces the non-autodecoded command to complete with a STALL handshake. It has no effect for autodecoded requests. No action Stall current USB command. Always read 0
4	Reserved		Reserved
3	DEV_CFG	0 1	If the USB device controller receives a SET_CONFIGURATION with a valid configuration value, and the device is in addressed state, it must write a 1 to the device configured (DEV_CFG) bit to inform the command decode that the device moves to configured state. The core sets the DEVSTAT.CFG bit to 1. If the device is already configured when the SET_CONFIGURATION request is received, the USB device controller does not have to set this bit. If the new configuration value is 0, USB device controller must set the SYSCON2.CLR_CFG bit to move to addressed state. Reading this bit always returns 0. Writing 0 into this bit has no effect. No action Allows DEVSTAT.CFG to be set. Always read 0
2	CLR_CFG	0 1	If the USB device controller receives a SET_CONFIGURATION with a configuration value of 0, and if the device is configured, it must write a 1 to the clear configured (CLR_CFG) bit to inform the command decode that the device becomes deconfigured (moves to addressed state). The core clears the DEVSTAT.CFG bit. Reading this bit always returns 0. Writing 0 into this bit has no effect. No action Allows DEVSTAT.CFG to be cleared. Always read 0
1 - 0	Reserved		Reserved

**28.3.1.9 Device Status Register (DEVSTAT)**

This read-only register provides a status reflecting the visible device states as defined in *USB1.1 Specification*. A write to this register has no effect.

**NOTE:** This register is double buffered. If IRQ\_EN.DS\_CHG\_IE is set (interrupt enabled), the background register is moved into foreground position only after clearing any pending DS\_CHG interrupt. Therefore, if there is a state change and a pending DS\_CHG interrupt has not been serviced yet, then the recent state change is not visible, because the background register was only updated and not moved into the foreground position.

The values depend on whether the reset action comes from the USB device controller (CPU) or USB host. They also depend on whether the device is attached or not when the USB device controller hardware reset event occurs.

**Figure 28-36. Device Status Register (DEVSTAT) [address = FCF78A10h]**

15				10			9	8
Reserved							B_HNP_ENABLE	A_HNP_SUPPORT
7		6	5	4	3	2	1	0
A_ALT_HNP_SUPPORT	R_WK_OK	USB_RESET	SUS	CFG	ADD	DEF	ATT	

LEGEND: R = Read, W = Write, n = value at reset



**Table 28-40. Device Status Register (DEVSTAT) Bit Field Descriptions**

Bit	Field	Value	Description
15 - 10	Reserved		Reserved
9	B_HNP_ENABLE	0 1	The HNP enable for B-device bit is used for On-The-Go feature selection purposes. See the <i>On-The-Go Supplement</i> to the <i>USB 2.0 Specification</i> . Not significant HNP enable for B-device Value after system reset or USB reset is low.
8	A_HNP_SUPPORT	0 1	The B-device is directly connected to an A-device port that supports HNP. This A_HNP_SUPPORT bit is used for On-The-Go feature selection purposes. See the <i>On-The-Go Supplement</i> to the <i>USB 2.0 Specification</i> . Not significant B-device connection to HNP capable A-device Value after system reset or USB reset is low.
7	A_ALT_HNP_SUPPORT	0 1	The B-device is connected to an A-device port that is not capable of HNP, but the A-device has an alternate port that is capable of HNP. This A_ALT_HNP_SUPPORT bit is used for On-The-Go feature selection purposes. See the <i>On-The-Go Supplement</i> to the <i>USB 2.0 Specification</i> . Not significant B-device connection to not HNP capable A-device Value after system reset or USB reset is low.
6	R_WK_OK	0 1	The remote wake-up granted bit is automatically set and cleared when the core receives a valid set/clear device feature request from the USB host. It returns a 1 when the USB host grants the function the ability to assert remote wake-up. Not significant Remote wake-up granted Value after system reset or USB reset is low
5	USB_RESET	0 1	USB reset signaling is active. The USB_RESET bit returns 1 when the USB host resets the USB bus. A valid USB reset resets the endpoint FIFOS, the other control register bits, except for SYSCON1.CFG_LOCK and the associated configuration registers (0x20 to 0x3F), IRQ_EN.DS_CHG_IE and IRQ_SRC.DS_CHG, and forces the device to the default state for DEVSTAT (this register). This bit is cleared at the end of reset. This bit, as well as other DEVSTAT bits, is double-buffered. If a pending interrupt has not been handled when a USB reset occurs and is handled only when USB reset is finished, the USB device controller does not see the USB_RESET bit going high and then low. Device is not being reset by USB host. Device is being reset by USB host. The value after the USB device controller hardware reset is low, and during USB reset is high (low after USB reset).
4	SUS	0 1	Suspended state bit: the device in suspended state is, at a minimum, attached to the USB, powered, has been reset by the USB host, and has not seen bus activity for 5 ms. It can also have a unique address and be configured for use. Because the device is suspended, however, the host cannot use the device function. This bit returns 1 when the USB device is in a suspended state. Not suspended Suspended Value after system reset or USB reset is low.
3	CFG	0 1	Configured state bit: the device is attached to the USB, powered, has been reset, has a unique address, and is configured. The host can use the function provided by the device. This bit returns 1 when the USB device has been configured after a set SYSCON2.DEV_CFG = 1. This bit remains set to 1 until the device becomes deconfigured. This bit is cleared when the core receives a valid SET_CONFIGURATION request and the USB device controller sets the SYSCON2.CLR_CFG bit. During the time this bit is not set to 1, transactions that are not for control EP0 are ignored. A GET_ENDPOINT_STATUS to a non-control endpoint is stalled. Not configured Configured Value after system reset or USB reset is low.

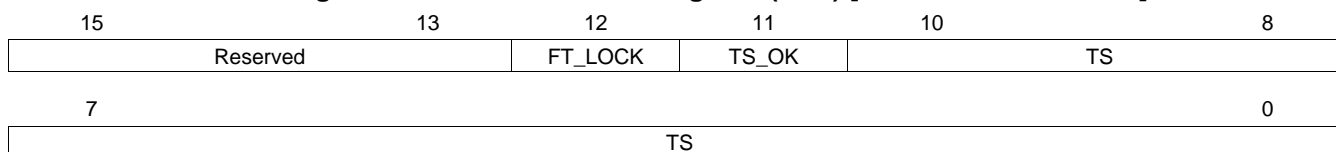


**Table 28-40. Device Status Register (DEVSTAT) Bit Field Descriptions (continued)**

Bit	Field	Value	Description
2	ADD	0 1	Addressed state bit: the device is attached to the USB, powered, has been reset, and a unique device address has been assigned. This bit (ADD) returns 1 after an ET_ADDRESS standard request. This bit remains set to 1 until the device becomes de-addressed. Not addressed Addressed Value after system reset or USB reset is low.
1	DEF	0 1	The default state bit returns 1 when the USB device is attached to the USB and powered, and has been reset. This bit remains set to 1 until the device becomes de-powered. The device moves into default state as soon as the USB reset is effective. Not in default Default The value after CPU is low, and after USB reset is high.
0	ATT	0 1	The attached state bit returns 1 when the device is attached to the USB and is powered. This bit remains set to 1 until the device powers down. Not attached Attached The value after USB device controller hardware reset is low (unattached) or high (attached), and after USB reset is high.

**28.3.1.10 Start of Frame Register (SOF)**

This read-only register provides a frame timer status for use in ISO communications. A write to this register is denied.

**Figure 28-37. Start of Frame Register (SOF) [address = FCF78A12h]**

LEGEND: R = Read, W = Write, n = value at reset

**Table 28-41. Start of Frame Register (SOF) Bit Field Descriptions**

Bit	Field	Value	Description
15 - 13	Reserved		Reserved
12	FT_LOCK	0 1	Frame timer locked bit: the USB host sends out a start of frame (SOF) packet every millisecond. When the device does not receive a SOF packet because of a bus error, a local start-of-frame that is used for ISO FIFO switch is generated. Once the core receives two valid SOF, separated by TF (time frame), it sets SOF.FT_LOCK to 1, only if TF is in frame interval IF allowed by the USB 1.1 specification (IF = [11964:12036] USB bit time). If TF is out of this interval, SOF.FT_LOCK value remains 0, and a local SOF is generated by the core. When SOF.FT_LOCK is set and the frame timer is locked to the timing TF, a local SOF is generated, if no valid SOF has been received in an interval of TF since the last valid SOF. The SOF.FT_LOCK bit is cleared if a valid SOF is received out of the interval IF. If the core receives a valid SOF in this interval, the frame timer locks to the new frame time. If the core does not receive a valid SOF, the frame timer remains locked to TF. When SOF.FT_LOCK is cleared, a local SOF is generated after the 12036 USB bit time, if no valid SOF has been received, and SOF.FT_LOCK remains 0. Frame timer is not locked. Frame timer is locked. The value after system reset or USB reset is low. Note: Each time the core receives a valid SOF out of the allowed interval IF, a local SOF is generated and the ISO FIFO switches.
11	TS_OK	0 1	The timestamp OK bit indicates that the timestamp in SOF.TS is valid for the current frame. It returns a 1 if a valid SOF packet was received from the USB host and a 0 otherwise. Timestamp is invalid. Timestamp is valid. Value after system reset or USB reset is low.

**Table 28-41. Start of Frame Register (SOF) Bit Field Descriptions (continued)**

Bit	Field	Value	Description
10 - 0	TS		The timestamp number field returns the timestamp from the last USB host-valid SOF packet. The frame number is valid if SOF.TS_OK is 1. In case of an SOF miss, this value is not updated and TS_OK is cleared. The value after system reset or USB reset is low (all 11 bits).

### 28.3.1.11 Interrupt Enable Register (IRQ\_EN)

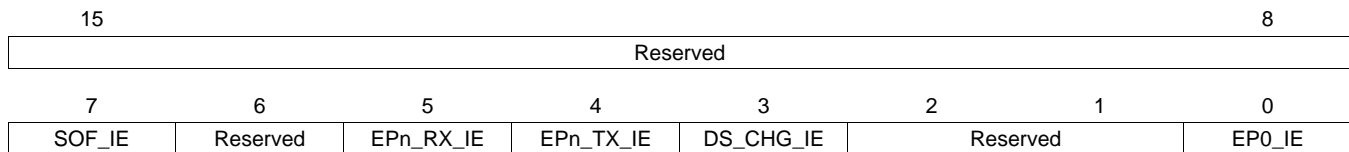
This read/write register enables all non-DMA interrupts (control, state changed, ISO, and non-ISO).

The values depend on whether the reset action comes from the USB device controller (CPU) or the USB host.

When the USB device controller sets a bit position to 1, an interrupt is signaled to the USB device controller if the corresponding IRQ\_SRC bit is asserted to 1 by the core for any IRQ\_SRC bit controlled by this bit. If reset to 0, the interrupt is masked and not signaled to the USB device controller.

- 0: Interrupt is disabled.
- 1: Interrupt is enabled.

The value after system reset or USB reset is low for all bits except IRQ\_EN.DS\_CHG\_IE bit, which remains unchanged after a reset from the USB host.

**Figure 28-38. Interrupt Enable Register (IRQ\_EN) [address = FCF78A14h]**


LEGEND: R = Read, W = Write, n = value at reset

**Table 28-42. Interrupt Enable Register (IRQ\_EN) Bit Field Descriptions**

Bit	Field	Value	Description
15 - 8	Reserved		Reserved
7	SOF_IE		Start of frame interrupt enable
6	Reserved		Reserved
5	EPn_RX_IE		Receive endpoint n interrupt enable (non-ISO)
4	EPn_TX_IE		Transmit endpoint n interrupt enable (non-ISO)
3	DS_CHG_IE		Device state changed interrupt enable
2 - 1	Reserved		Reserved
0	EP0_IE		EP0 transactions interrupt enable

### 28.3.1.12 DMA Interrupt Enable Register (DMA\_IRQ\_EN)

This read/write register enables all DMA interrupts.

When the USB device controller sets a bit position to 1, an interrupt is signaled to the USB device controller if the corresponding bit in the IRQ\_SRC register is asserted to 1 by the core. If reset to 0, the interrupt is masked and not signaled to the USB device controller.

- 0: Interrupt is disabled.
- 1: Interrupt is enabled.

The value after system reset or USB reset is low for all bits.

**Figure 28-39. DMA Interrupt Enable Register (DMA\_IRQ\_EN) [address = FCF78A16h]**

15	Reserved				11	10	9	8
					TX2_DONE_IE	RX2_CNT_IE	RX2_EOT_IE	
7	6	5	4	3	2	1	0	
Reserved	TX1_DONE_IE	RX1_CNT_IE	RX1_EOT_IE	Reserved	TX0_DONE_IE	RX0_CNT_IE	RX0_EOT_IE	

LEGEND: R = Read, W = Write, n = value at reset

**Table 28-43. DMA Interrupt Enable Register (DMA\_IRQ\_EN) Bit Field Descriptions**

Bit	Field	Value	Description
15 - 11	Reserved		Reserved
10	TX2_DONE_IE		Transmit DMA channel 2 done interrupt enable
9	RX2_CNT_IE		Receive DMA channel 2 transactions count interrupt enable
8	RX2_EOT_IE		Receive DMA channel 2 end of transfer interrupt enable
7	Reserved		Reserved
6	TX1_DONE_IE		Transmit DMA channel 1 done interrupt enable
5	RX1_CNT_IE		Receive DMA channel 1 transactions count interrupt enable
4	RX1_EOT_IE		Receive DMA channel 1 end of transfer interrupt enable
3	Reserved		Reserved
2	TX0_DONE_IE		Transmit DMA channel 0 done interrupt enable
1	RX0_CNT_IE		Receive DMA channel 0 transactions count interrupt enable
0	RX0_EOT_IE		Receive DMA channel 0 end of transfer interrupt enable

**28.3.1.13 Interrupt Source Register (IRQ\_SRC)**

This read/clear only register identifies and clears the source of the interrupt signaled by a set flag.

The value depends on whether the reset action comes from the USB device controller (CPU) or the USB host.

The USB device controller can clear a set bit location only by writing a 1 into the bit location (except for the setup bit, which is automatically cleared by the core). A write 0 has no effect.

When the core sets a bit location to 1, an interrupt is signaled to the USB device controller if the interrupt was enabled.

- 0: No interrupt
- 1: Interrupt signaled

The value after the system reset or USB reset is low except for the IRQ\_SRC.DS\_CHG bit, which is high after a USB reset.

**Figure 28-40. Interrupt Source Register (IRQ\_SRC) [address = FCF78A18h]**

15	Reserved				11	10	9	8
					TXn_DONE	RXn_CNT	RXn_EOT	
7	6	5	4	3	2	1	0	
SOF	Reserved	EPn_RX	EPn_TX	DS_CHG	SETUP	EP0_RX	EP0_TX	

LEGEND: R = Read, W = Write, n = value at reset

**Table 28-44. Interrupt Source Register (IRQ\_SRC) Bit Field Descriptions**

Bit	Field	Value	Description
15 - 11	Reserved		Reserved
10	TXn_DONE	0 No action 1 Non-ISO transmit DMA transfer for a channel has ended. The value after system reset or USB reset is low.	The transmit DMA channel n done interrupt flag (non-ISO) bit is only for non-ISO DMA transfer. This bit is never set for ISO DMA transfer. The core automatically sets this bit when a transmit DMA channel has completed the programmed transfer by servicing the last IN transaction from the USB host. This is when TXDMA <sub>n</sub> .TXn_TSC (transfer size counter) equals 0, and the last IN transaction completes with an ACK. When this bit is asserted, the USB device controller must read the DMAN_STAT register to identify the endpoint number for which the transfer completed. The endpoint interrupt IRQ_SRC.EPn_TX is never set for the assigned endpoint to TX DMA channel n.
9	RXn_CNT	0 No action 1 Non-ISO receive DMA transfer for a channel has reached transactions count level. The value after system reset or USB reset is low.	The receive DMA channel n transactions count interrupt flag (non-ISO) bit is only for non-ISO DMA transfer. This bit is never set for ISO DMA transfer and never set if the interrupt enable bit associated with it is not set. It means that no poll operation is possible on the DMA. The core automatically sets this bit during an active Receive DMA transfer each time RXDMA <sub>n</sub> .RXn_TC equals 0 after an OUT transaction with ACK status. This bit is set after the RX DMA data have been read (end of DMA request). When this bit is asserted, the USB device controller must read DMAN_STAT register to identify the endpoint number for which the transfer completed. An RXn_CNT IT is asserted also if RXDMA <sub>n</sub> .RXn_STOP is set; in this case, both IRQ_SRC.RXn_EOT and IRQ_SRC.RXn_CNT are asserted.
8	RXn_EOT	0 No action 1 Non-ISO receive DMA transfer for a channel has ended. The value after system reset or USB reset is low.	The receive DMA channel n end of transfer interrupt flag (non-ISO) bit is for non-ISO DMA transfer only. This bit is never set for ISO DMA transfer and never set if the interrupt enable bit associated with it is not set. It means that no poll operation is possible on the DMA. THE core automatically sets this bit when a receive DMA channel detects an end-of-transfer (EOT) packet during the last OUT transaction from the USB host. This bit is set after the RX DMA data have been read (end of DMA request). When this happens, the DMA assigned endpoint FIFO is kept disabled (STAT_FLG.FIFO_EN = 0) to avoid receiving a new packet data from the USB host. The USB device controller can grant DMA transfer again to the same endpoint by enabling the FIFO again (STAT_FLG.FIFO_EN = 1).An end of transfer is detected when the core receives a data packet that is less than the configured endpoint FIFO size (or is empty), or when RXDMA <sub>n</sub> .RXn_TC equals 0 after an OUT transaction with ACK status and the RXDMA <sub>n</sub> .RXn_STOP bit is set. When this bit is asserted, the USB device controller must read DMAN_STAT.DMA <sub>n</sub> _RX_IT_SRC to identify the endpoint number for which the transfer completed, and DMAN_STAT.DMA <sub>n</sub> _RX_SB must be informed of an odd number of bytes received during the last transaction (useful for 16-bit read access from the DATA_DMA register).The endpoint interrupt IRQ_SRC.EPn_RX is never set to RX DMA channel for the assigned endpoint.
7	SOF	0 No action 1 Start-of-frame packet received (or internal SOF) The value after system reset or USB reset is low.	Start of frame interrupt flag bit: every millisecond, the USB host outputs a start-of-frame packet to the functions. The SOF bit reflects when a new SOF is received. Writing a 1 in the SOF bit location clears the flag. Writing a 0 has no effect. In accordance with the USB1.1 specification, if SOF is received corrupted or is not received, the core still sets this flag at the same rate (if bit SOF.FT_LOCK = 1) or after 12043 USB bit time (if bit SOF.FT_LOCK = 0).
6	Reserved		Reserved
5	EPn_RX	0 No action 1 OUT transaction detected on an endpoint The value after system reset or USB reset is low.	The EPn OUT transactions interrupt flag bit only concerns non-ISO endpoints. The core automatically sets the EPn_RX bit when a handshake sequence occurs for an OUT transaction to an interrupt or bulk endpoint (NAK with SYSCON1.NAK_EN set, ACK or STALL). The USB device controller must read the EPn_STAT register to identify the endpoint causing the interrupt.

**Table 28-44. Interrupt Source Register (IRQ\_SRC) Bit Field Descriptions (continued)**

Bit	Field	Value	Description
4	EPn_TX	0 1	The EPn IN transactions interrupt flag bit only concerns non-ISO endpoints. The core automatically sets this bit when a handshake sequence occurs for an IN transaction to an interrupt or bulk endpoint (NAK with SYSCON1.NAK_EN set, ACK or STALL). The USB device controller must read the EPn_STAT register to identify the endpoint causing the interrupt.  No action IN transaction detected on an endpoint value after system reset or USB reset is low.
3	DS_CHG	0 1	Device state changed interrupt flag bit: the core automatically resets the DS_CHG bit when the state of the device changes. This is when the core modifies any of the bits present in the DEVSTAT register. When this bit is cleared, the background DEVSTAT register moves into foreground position.  No action Device state change detected value after USB device controller hardware reset is low and after USB reset is high.
2	SETUP	0 1	Setup transaction interrupt flag bit: the core automatically sets the SETUP bit when a valid setup transaction completes on control endpoint for a non-autodecoded control request and automatically clears it when the USB device controller sets EP_NUM.SETUP_SEL bit when reading setup data. A write 1 to it has no effect.  No action Valid setup transaction occurred on endpoint 0. Value after system reset or USB reset is low.
1	EP0_RX	0 1	EP0 OUT transactions interrupt flag bit: the core automatically sets the EP0_RX bit when a handshake sequence occurs for a non-autodecoded OUT transaction to control endpoint (NAK with SYSCON1.NAK_EN set, ACK or STALL).  No action OUT transaction on EP0 Value after system reset or USB reset is low.
0	EP0_TX	0 1	EP0 IN transactions interrupt flag bit: the core automatically sets this bit when a handshake sequence occurs for a non-autodecoded IN transaction to control endpoint (NAK with SYSCON1.NAK_EN set, ACK or STALL).  No action IN transaction on EP0 Value after system reset or USB reset is low.

**28.3.1.14 Non-ISO Endpoint Interrupt Status Register (EPN\_STAT)**

This read-only register identifies the non-ISO endpoint causing an EPn interrupt. A write into it is denied.

**NOTE:** If a nontransparent transaction occurs before a previous one on another endpoint in the same direction that has been handled, the second interrupt is asserted only after the USB device controller clears the first one and EPN\_STAT is updated with the corresponding interrupt assertion.

**Figure 28-41. Non-ISO Endpoint Interrupt Status Register (EPN\_STAT) [address = FCF78A1Ah]**

15	12	11	8
Reserved		EPn_RX_IT_SRC	
7	4	3	0
Reserved		EPn_TX_IT_SRC	

LEGEND: R = Read, W = Write, n = value at reset

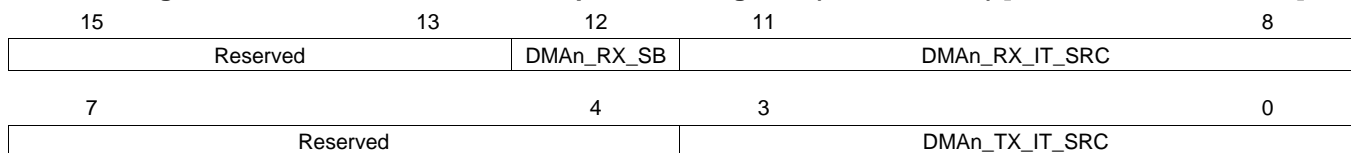
**Table 28-45. Non-ISO Endpoint Interrupt Status Register (EPN\_STAT) Bit Field Descriptions**

Bit	Field	Value	Description
15 - 12	Reserved		Reserved
11 - 8	EPn_RX_IT_SRC	0000 0001 ... 1111	The receive endpoint interrupt source (non-ISO) bit only concerns non-ISO endpoints. When the IRQ_SRC.EPn_RX flag is set, the endpoint causing the interrupt condition is encoded in these four register bits. When the IRQ_SRC.EPn_RX flag is cleared, the four bits read as 0. No receive endpoint interrupt is pending. EP1 EP15 Values after system reset or USB reset are low (all four bits).
7 - 4	Reserved		Reserved
3 - 0	EPn_TX_IT_SRC	0000 0001 ... 1111	Transmit endpoint interrupt source (non-ISO) bit only concerns non-ISO endpoints. When the IRQ_SRC.EPn_TX flag is set, the endpoint that causes this flag to be set is encoded in these four register bits. When the IRQ_SRC.EPn_TX flag is cleared, the four bits read as 0. No transmit endpoint interrupt is pending. EP1 EP15 Values after system reset or USB reset are low (all 4 bits).

**28.3.1.15 Non-ISO DMA Interrupt Status Register (DMAN\_STAT)**

This read-only register identifies the endpoint that causes a DMA interrupt. A write into it is denied.

**NOTE:** If a DMA interrupt occurs before a previous one on another endpoint in the same direction that has been handled, the second interrupt is asserted only after the USB device controller clears the first one. DMAN\_STAT is updated when the corresponding interrupt is asserted.

**Figure 28-42. Non-ISO DMA Interrupt Status Register (DMAN\_STAT) [address = FCF78A1Ch]**


LEGEND: R = Read, W = Write, n = value at reset

**Table 28-46. Non-ISO DMA Interrupt Status Register (DMAN\_STAT) Bit Field Descriptions**

Bit	Field	Value	Description
15 - 13	Reserved		Reserved
12	DMAn_RX_SB	0 1	The DMA receive single byte (non-ISO) bit only concerns non-ISO endpoints (ISO endpoints receive a constant number of bytes). This bit is set when an IRQ_SRC.RXn_EOT interrupt is asserted and the core received an odd number of bytes during the last transaction. It is used to know the exact number of bytes received in case of a 16-bit read access from DATA_DMA register. When the IRQ_SRC.RXn_EOT flag is cleared, this bit reads as 0. No EOT DMA interrupt is pending or the core received an even number of bytes during the last transaction. An EOT DMA interrupt is pending and an odd number of bytes was received during the last transaction. Value after system reset or USB reset is low.

**Table 28-46. Non-ISO DMA Interrupt Status Register (DMAN\_STAT) Bit Field Descriptions (continued)**

Bit	Field	Value	Description
11 - 8	DMAn_RX_IT_SRC	0000 0001 ... 1111	The DMA receive interrupt source (non-ISO) bit only concerns non-ISO endpoints. When the IRQ_SRC.EPn_RX flag is set, the endpoint causing this flag to be set is encoded in these four register bits. When the IRQ_SRC.EPn_RX flag is cleared, the four bits read as 0. No receive DMA interrupt is pending. EP1 ... EP15 Values after system reset or USB reset are low (all 4 bits).
7 - 4	Reserved		Reserved
3 - 0	DMAn_TX_IT_SRC	0000 0001 ... 1111	The DMA transmit interrupt source (non-ISO) bit only concerns non-ISO endpoints. When the IRQ_SRC.EPn_TX flag is set, the endpoint that causes this flag to be set is encoded in these four register bits. When the IRQ_SRC.EPn_TX flag is cleared, the four bits read as 0. No transmit DMA interrupt is pending. EP1 ... EP15 Values after system reset or USB reset are low (all 4 bits).

**28.3.1.16 DMA Receive Channels Configuration Register (RXDMA\_CFG)**

This read/write register enables the three possible DMA receive channels and selects the endpoint number that is assigned to each of these DMA channels. An endpoint used by an RX DMA channel must have been configured (through register EPn\_RX). The RXDMA\_CFG register can be filled when SYSCON1.CFG\_LOCK is set.

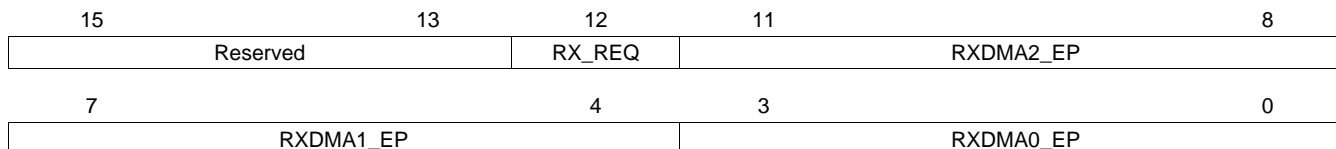
Only one channel is serviced at a time, so it is better to configure ISO endpoints on the first channel (\_EP). If the ISO endpoint is configured on channel 2, it can never be serviced with low software and a fast USB host.

The USB device controller transmit endpoint DMA channels 0, 1, and 2 are associated with device controller inputs.

---

**NOTE:** System software must not program RXDMAx\_EP to endpoint numbers that are not configured as DMA endpoints.

---

**Figure 28-43. DMA Receive Channels Configuration Register (RXDMA\_CFG) [address = FCF78A20h]**

LEGEND: R = Read, W = Write, n = value at reset

**Table 28-47. DMA Receive Channels Configuration Register (RXDMA\_CFG) Bit Field Descriptions**

Bit	Field	Value	Description
15 - 13	Reserved		Reserved
12	RX_REQ	0 1	The RX DMA request active level or pulse bit allows the RXDMA request to be configurable level or pulse-sensitive. When pulse-sensitive, the request is active during 2 cycles. RX DMA request active level RX DMA request active pulse Value after system reset or USB reset is low.



**Table 28-47. DMA Receive Channels Configuration Register (RXDMA\_CFG) Bit Field Descriptions (continued)**

Bit	Field	Value	Description
11 - 8	RXDMA2_EP	0000 0001 ... 1111	Receive endpoint number for DMA channel 2. The endpoint number binary encoded in these four bits is the current receive endpoint selected for DMA channel 2. A zero value indicates that the DMA channel 2 is deactivated. Any other value automatically enables receive DMA transfer for the selected endpoint.  Receive DMA channel 2 is deactivated. EP1  EP15 Values after system reset or USB reset are low (all 4 bits).
7 - 4	RXDMA1_EP	0000 0001 ... 1111	Receive endpoint number for DMA channel 1. The endpoint number binary encoded in these four bits is the current receive endpoint selected for DMA channel 1. A zero value indicates that the DMA channel 1 is deactivated. Any other value automatically enables receive DMA transfer for the selected endpoint.  Receive DMA channel 1 is deactivated. EP1  EP15 Values after system reset or USB reset are low (all 4 bits).
3 - 0	RXDMA0_EP	0000 0001 ... 1111	Receive endpoint number for DMA channel 0. The endpoint number binary encoded in these four bits is the current receive endpoint selected for DMA channel 0. A zero value indicates that the DMA channel 0 is deactivated. Any other value automatically enables receive DMA transfer for the selected endpoint.  Receive DMA channel 0 is deactivated. EP1  EP15 Values after system reset or USB reset are low (all 4 bits).

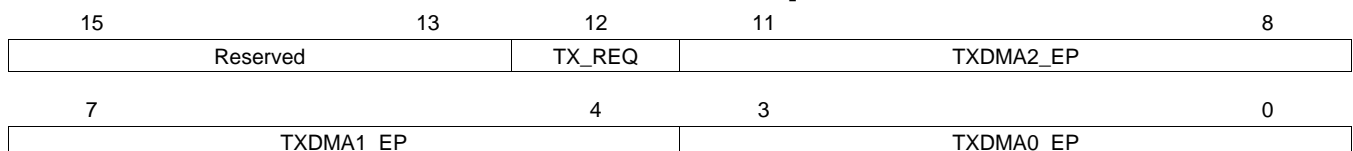
**28.3.1.17 DMA Transmit Channels Configuration Register (TXDMA\_CFG)**

This read/write register enables the three possible DMA transmit channels and selects the endpoint number that is assigned to each. An endpoint used by a TX DMA channel must have been configured (through register EPn\_TX). TXDMA\_CFG register can be filled when SYSCON1.CFG\_LOCK is set.

Only one channel is serviced at a time, so it is better to configure ISO endpoints on the first channel (TXDMA0\_EP). If, the/your ISO endpoint is configured on the channel 2 and it can never be serviced with low software and a fast USB host.

USB device controller transmit endpoint DMA channels 0, 1, and 2 are associated with device controller inputs.

**Figure 28-44. DMA Transmit Channels Configuration Register (TXDMA\_CFG) [address = FCF78A22h]**



LEGEND: R = Read, W = Write, n = value at reset

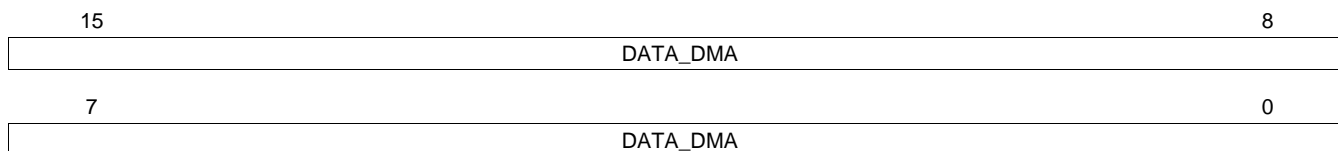


**Table 28-48. DMA Transmit Channels Configuration Register (TXDMA\_CFG) Bit Field Descriptions**

Bit	Field	Value	Description
15 - 13	Reserved		Reserved
12	TX_REQ	0 1	The TX DMA request active level or pulse bit allows the TXDMA request to be configurable level or pulse-sensitive. When pulse-sensitive, the request is active for two cycles. TX DMA request active level TX DMA request active pulse value after system reset or USB reset is low.
11 - 8	TXDMA2_EP	0000 0001 ... 1111	Transmit endpoint number for DMA channel 2: the endpoint number binary encoded in these four bits is the current transmit endpoint selected for DMA channel 2. A zero value indicates that the DMA channel 2 is deactivated. Any other value automatically enables transmit DMA transfer for the selected endpoint. Transmit DMA channel 2 is deactivated. EP1 EP15 Values after system reset or USB reset are low (all four bits).
7 - 4	TXDMA1_EP	0000 0001 ... 1111	Transmit endpoint number for DMA channel 1: the endpoint number binary encoded in these four bits is the current transmit endpoint selected for DMA channel 1. A zero value indicates that the DMA channel 1 is deactivated. Any other value automatically enables transmit DMA transfer for the selected endpoint. Transmit DMA channel 1 is deactivated. EP1 EP15 Values after system reset or USB reset are low (all four bits).
3 - 0	TXDMA0_EP	0000 0001 ... 1111	Transmit endpoint number for DMA channel 0: the endpoint number binary encoded in these four bits is the current transmit endpoint selected for DMA channel 0. A zero value indicates that the DMA channel 0 is deactivated. Any other value automatically enables transmit DMA transfer for the selected endpoint. Transmit DMA channel 0 is deactivated. EP1 EP15 Values after system reset or USB reset are low (all four bits).

**28.3.1.18 DMA FIFO Data Register (DATA\_DMA)**

This register is the entry point to write or to read data into/from an endpoint used in a DMA transfer through DMA channel 0, 1 or 2.

**Figure 28-45. DMA FIFO Data Register (DATA\_DMA) [address = FCF78A24h]**

LEGEND: R = Read, W = Write, n = value at reset

**Table 28-49. DMA FIFO Data Register (DATA\_DMA) Bit Field Descriptions**

Bit	Field	Value	Description
15 - 0	DATA_DMA		<p>DMA FIFO data. When an RX DMA request is active for a channel (only one active at a given time), this register contains the data that the core received from the USB host OUT transaction using this channel. Data can be accessed by the main DMA controller engine (read access) in response to the DMA request for the channel. When a TX DMA request is active for a channel (only one active at a given time), this register contains the data written by the main DMA controller engine (write access) in response to a DMA request for the transmit channel to be sent to the USB host during the next IN transaction.</p> <p>Note: It is possible for both an RX DMA request and a TX DMA request to be active at the same time. In this case, the main DMA controller engine can access both transmit endpoint and receive endpoint FIFO. A read access to DATA_DMA register affects the endpoint that caused the RX DMA request to be active, and a write access affects the endpoint that caused the TX DMA request to be active.</p> <p>Note: The USB device controller must not access this register directly; however, there is no hardware mechanism to prevent such access. No access into this register must be made out of DMA request handling.</p>

**28.3.1.19 Transmit DMA Control Register n (TXDMA<sub>n</sub>)**

This read/write register controls the operation of the transmit DMA channel n (n = 0, 1, 2).

**Figure 28-46. Transmit DMA Control Register n (TXDMA<sub>n</sub>) [address = FCF78A28h — FCF78A2Ch]**

15	14	13	10	9	8
TX <sub>n</sub> _EOT	TX <sub>n</sub> _START	Reserved		TX <sub>n</sub> _TSC	
					7
					0
TX <sub>n</sub> _TSC					

LEGEND: R = Read, W = Write, n = value at reset

**Table 28-50. Transmit DMA Control Register n (TXDMA<sub>n</sub>) Bit Field Descriptions**

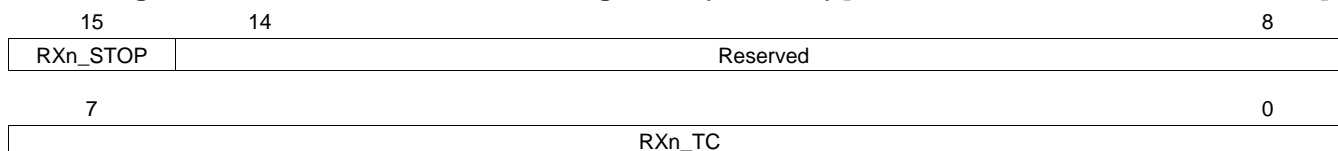
Bit	Field	Value	Description
15	TX <sub>n</sub> _EOT	<p>0 DMA transfer size is in buffers.</p> <p>1 DMA transfer size is in bytes.</p> <p>Value after system reset or USB reset is low.</p>	<p>Transmit DMA channel n end of transfer: the TX<sub>n</sub>_EOT bit can be either 0 or 1 for BULK DMA transfer. When the USB device controller sets it to 1, this bit signals the core that the transfer size set in TXDMA<sub>n</sub>.TX<sub>n</sub>_TSC is in bytes. A TX one interrupt (IRQ_SRC.TX<sub>n</sub>_DONE) is asserted with the last IN transaction. If the number of bytes set in TXDMA<sub>n</sub>.TX<sub>n</sub>_TSC is a multiple of the endpoint buffer size, the TX done interrupt is asserted only after an IN transaction with an empty data packet. When cleared, the transfer size set in TX<sub>n</sub>_TSC is in full buffer size for the endpoint selected (BULK only). A TX done interrupt is asserted when the last buffer is sent with the last IN transaction. This mode is to be used for a partial bulk transfer of a large file exceeding 1023 bytes.</p>
14	TX <sub>n</sub> _START	<p>0 No action</p> <p>1 DMA transfer start Always reads 0</p>	<p>Transmit DMA channel n start. The USB device controller sets this bit to tell the device that the main DMA system is ready to transmit the number of bytes or buffers. Once set, the DMA transfer cannot be interrupted, unless the USB device controller clears the endpoint in the TXDMA_CFG register. A write 0 into this bit has no effect and a read to this bit always returns 0. The IRQ_SRC.TX<sub>n</sub>_DONE interrupt bit is asserted when the DMA transfer ends.</p>
13 - 10	Reserved		Reserved

**Table 28-50. Transmit DMA Control Register n (TXDMA<sub>n</sub>) Bit Field Descriptions (continued)**

Bit	Field	Value	Description
9 - 0	TXn_TSC		<p>Transmit DMA channel n transfer size counter. The binary encoded value from 0 to 1023, which the USB device controller writes into this register, corresponds to the number of bytes or number of buffer transfers (function of TXDMA<sub>n</sub>.TXn_EOT) to be transmitted by the transmit DMA channel n. When read, the register reflects the number of bytes/buffers the USB device has still to transmit. Read mode is only provided for software debug purposes.</p> <p>Note: For ISO transfer, the user must verify that the set value does not exceed the ISO FIFO size for the endpoint. There is no hardware mechanism to prevent this situation. If this situation occurs, results are unpredictable.</p> <p>Note: For bulk transfer, when TXDMA<sub>n</sub>.TXn_EOT = 0, a set value of TXDMA<sub>n</sub>.TXn_TSC = 0 means 1024 buffers and not 0. The counter then operates in the following way: 000, 3FF, 3FE, 0001, 000, stop. When TXDMA<sub>n</sub>.TXn_EOT = 1, a set value of TXDMA<sub>n</sub>.TXn_TSC = 0 a NULL packet is sent in response to the next IN token.</p> <p>Values after system reset or USB reset are low (all 10 bits).</p>

**28.3.1.20 Receive DMA Control Register n (RXDMA<sub>n</sub>)**

This read/write register monitors incoming OUT transactions during DMA transfer on channel n (n=0,1,2).

**Figure 28-47. Receive DMA Control Register n (RXDMA<sub>n</sub>) [address = FCF78A30h — FCF78A34h]**

LEGEND: R = Read, W = Write, n = value at reset

**Table 28-51. Receive DMA Control Register n (RXDMA<sub>n</sub>) Bit Field Descriptions**

Bit	Field	Value	Description
15	RXn_STOP		<p>Receive DMA channel n transfer stop. When this bit is set, an RXn_EOT interrupt is asserted to the USB device controller after n OUT transactions where n is the encoded binary value + 1 programmed into RXDMA<sub>n</sub>.RXn_TC field. This register is used when no smaller than buffer size packet is received at an end-of-transfer (EOT), and the USB device controller expects a given amount of data for the transfer.</p> <p>Note: At end of transfer, the DMA channel is disabled and all OUT transactions received to the assigned endpoint are sent NAK by the core. The USB device controller must set CTRL.SET_FIFO_EN for the endpoint to reenables the channel.</p> <p>Values after system reset or USB reset are low.</p>
14 - 8	Reserved		Reserved
7 - 0	RXn_TC		<p>Receive DMA channel n transactions count. The USB device controller can ask for an interrupt each n OUT transactions where n is the encoded binary value + 1 programmed into RXDMA<sub>n</sub>.RXn_TC field. This register must be programmed to the desired transaction watermark limit prior to enabling the DMA transfer for the receive DMA channel n.</p> <p>Note: A reached watermark does not disable an active DMA transfer if RXDMA<sub>n</sub>.RXn_STOP was not set. If RXDMA<sub>n</sub>.RXn_STOP was set for the transfer both RXn_CNT and RXn_EOT interrupts are asserted. A read to that register returns the number of transactions remaining before the IRQ_SRC.RXn_CNT interrupt flag is asserted. This read mode is only provided for software debug purposes.</p> <p>Values after system reset or USB reset are low (all 8 bits).</p>

**28.3.1.21 Endpoint 0 Configuration Register (EP0)**

This read/write register gives the device configuration for control endpoint 0.

Values depend on whether the reset action comes from USB device controller (CPU) or the USB host.

**Figure 28-48. Endpoint 0 Configuration Register (EP0) [address = FCF78A40h]**

15	14	13	12	11	10	8
Reserved		EP0_SIZE		Reserved		EP0_PTR
						7
						0
EP0_PTR						

LEGEND: R = Read, W = Write, n = value at reset

**Table 28-52. Endpoint 0 Configuration Register (EP0) Bit Field Descriptions**

Bit	Field	Value	Description
15 - 14	Reserved		Reserved
13 - 12	EP0_SIZE	00 8 bytes 01 16 bytes 10 32 bytes 11 64 bytes Values after USB device controller hardware reset or USB reset are unchanged (which means that value is unknown until first write access).	Endpoint 0 FIFO size. This field contains the endpoint 0 FIFO size value and must match the value sent by the USB device controller to the USB host during the GET_DEVICE_DESCRIPTOR request preceding configuration phase. Status flags (STAT_FLG.NON_ISO_FIFO_EMPTY and STAT_FLG.NON_ISO_FIFO_FULL) and overrun and underrun conditions are based on this value for all IN and OUT transactions to endpoint 0. The USB device controller must fill this field before setting SYSCON1.CFG_LOCK.
11	Reserved		Reserved
10 - 0	EP0_PTR	0x000 Address = BASE (not permitted) 0x001 Address = BASE + 8 bytes 0x002 Address = BASE + 16 bytes 0x003 Address = BASE + 24 bytes ... 0x0FF Address = BASE + 2040 bytes Values after USB device controller hardware reset or USB reset are unchanged (which means that values are unknown until first write access).	Endpoint 0 pointer. This field contains the address of the endpoint 0 pointer. Value 0x000 is not permitted (reserved for setup FIFO). Note: The pointer value must be set to a value < 0xFF, because the memory size is 2K bytes and a pointer coded value = 0xFF corresponds to 2040 bytes. Addressing upper bytes results in memory overlap.

### 28.3.1.22 Receive Endpoint n Configuration Register (EPn\_RX)

This read/write register gives the device configuration for non-control receive endpoint n (1:15). The endpoints size fields must match values sent by the USB device controller to the USB host in response to the GET\_CONFIGURATION\_DESCRIPTOR during configuration phase.

The USB device controller must fill this field before setting SYSCON1.CFG\_LOCK and must not change the values once SYSCON1.CFG\_LOCK is set.

Values depend on whether the reset action comes from USB device controller (CPU) or USB host.

**Figure 28-49. Receive Endpoint n Configuration Register (EPn\_RX) [address = FCF78A42h — FCF78A5Eh]**

15	14	13	12	11	10	8
EPn_RX_VALID	EPn_RX_SIZE/DB	EPn_RX_SIZE		EPn_RX_ISO	EPn_RX_PTR	
						7
						0
EPn_RX_PTR						

LEGEND: R = Read, W = Write, n = value at reset

**Table 28-53. Receive Endpoint n Configuration Register (EPn\_RX) Bit Field Descriptions**

Bit	Field	Value	Description
15	EPn_RX_VALID	0 1	The receive endpoint n valid bit must be set by the USB device controller to allow receive endpoint n to be used for USB transfers as part of the device configuration. If not set, all transactions to this endpoint are ignored by the core. Receive endpoint n does not exist for this configuration. Receive endpoint n is part of the device configuration. Values after USB device controller hardware reset are low, after USB reset is unchanged.
14	EPn_RX_SIZE/DB	0 1	Receive non-ISO (or ISO) endpoint n double-buffer (DB). This bit is only for non-ISO endpoints. For ISO endpoints, which are always double-buffered, this bit is endpoint size MSB. This bit must be set by the USB device controller to allow double buffering for receive non-ISO endpoint n. This is used to reduce number of transactions resulting in NAK handshake. No double buffer for non-ISO receive endpoint n. Double buffer used for non-ISO receive endpoint n. Values after USB device controller hardware reset or USB reset are unchanged (which means that values are unknown until first write access).
13 - 12	EPn_RX_SIZE	00 000 01 001 10 010 11 011 100 101 110 111	The receive endpoint n size field contains the endpoint n FIFO size value. Status flags (STAT_FLG.NON_ISO_FIFO_EMPTY, STAT_FLG.NON_ISO_FIFO_FULL, STAT_FLG.ISO_FIFO_EMPTY, and STAT_FLG.ISO_FIFO_FULL) and overrun and underrun conditions are based on this value for all OUT transactions to endpoint n. Non-ISO 8 bytes ISO 8 bytes [13:12] 16 bytes [14:12] 16 bytes 32 bytes 32 bytes 64 bytes 64 bytes 128 bytes 256 bytes 512 bytes 1023 bytes Note: For ISO endpoints, a size of 1023 bytes takes up the whole memory and prevents it from being programmed with a 2K-byte USB device controller. Values after USB device controller hardware reset or USB reset are unchanged (which means that values are unknown until first write access).
11	EPn_RX_ISO	0 1	The receive ISO endpoint n field must be set if the receive endpoint n type is isochronous in the desired device configuration. If not set, the endpoint type is bulk or interrupt (the hardware does not distinguish bulk type from interrupt). Receive endpoint n type is bulk or interrupt. Receive endpoint n type is isochronous. Values after USB device controller hardware reset or USB reset are unchanged.

**Table 28-53. Receive Endpoint n Configuration Register (EPn\_RX) Bit Field Descriptions (continued)**

Bit	Field	Value	Description
10 - 0	EPn_RX_PTR		The receive endpoint n pointer field contains the address of the receive endpoint n pointer. Value 0x000 is not permitted (reserved for setup FIFO). Note: For ISO endpoints or for non-ISO endpoints that allow double-buffering, 2*RX buffer size must be reserved for ping-pong.
		0x000	Address = BASE (not permitted)
		0x001	Address = BASE + 8 bytes
		0x002	Address = BASE + 16 bytes
		0x003	Address = BASE + 24 bytes
		...	
		0x0FF	Address = BASE + 2040 bytes Values after USB device controller hardware reset or USB reset are unchanged (which means that values are unknown until first write access). Note: Pointer value must be set to a value < 0xFF, because the memory size is 2K bytes and a pointer coded value = 0xFF corresponds to 2040 bytes. Addressing upper bytes results in memory overlap.

### 28.3.1.23 Transmit Endpoint n Configuration Register (EPn\_TX)

This read/write register gives the device configuration for non-control transmit endpoint n (1:15). The endpoints size fields must match the values sent by the USB device controller to the USB host in response to the GET\_CONFIGURATION\_DESCRIPTOR during configuration phase.

**NOTE:** The USB device controller must fill this field before setting SYSCON1.CFG\_LOCK and must not change the values once SYSCON1.CFG\_LOCK is set.

**Figure 28-50. Transmit Endpoint n Configuration Register (EPn\_TX) [address = FCF78A62h — FCF78A7Eh]**

15	14	13	12	11	10	8
EPn_TX_VALID	EPn_TX_SIZE/DB	EPn_TX_SIZE		EPn_TX_ISO	EPn_TX_PTR	
7						0
EPn_TX_PTR						

LEGEND: R = Read, W = Write, n = value at reset

**Table 28-54. Transmit Endpoint n Configuration Register (EPn\_TX) Bit Field Descriptions**

Bit	Field	Value	Description
15	EPn_TX_VALID		The transmit endpoint n valid bit must be set by the USB device controller to allow transmit endpoint n to be used for USB transfers as part of the device configuration. If not set, all transactions to this endpoint are ignored by the core.
		0	Transmit endpoint n does not exist for this configuration.
		1	Transmit endpoint n is part of the device configuration. Values after USB device controller hardware reset are low, after USB reset is unchanged.
14	EPn_TX_SIZE/DB		Transmit non-ISO (or ISO) endpoint n double buffer. This bit is only for non-ISO endpoints used in DMA mode. For ISO endpoints, which are always double buffered, this bit is endpoint size MSB. For non-ISO endpoints that are not used in a DMA transfer, double buffering is not provided. This bit must be set by the USB device controller to allow double buffering for transmit non-ISO endpoint n, when used in a DMA transfer. This is used to reduce the number of transactions resulting in NAK handshake.
		0	No double buffer for non-ISO transmit endpoint n.
		1	Double buffer used for non-ISO transmit endpoint n. Values after system reset or USB reset is unchanged. Or transmit ISO endpoint n size[2]
13 - 12	EPn_TX_SIZE		Transmit endpoint n size

**Table 28-54. Transmit Endpoint n Configuration Register (EPn\_TX) Bit Field Descriptions (continued)**

Bit	Field	Value	Description
11	EPn_TX_ISO	0 1	<p>The transmit ISO endpoint n field must be set if the transmit endpoint n type is isochronous in the desired device configuration. If not set, the endpoint type is bulk or interrupt (the hardware does not distinguish bulk type from interrupt).</p> <p>0 Transmit endpoint n type is bulk or interrupt. 1 Transmit endpoint n type is isochronous. Value after USB reset is unchanged.</p>
10 - 0	EPn_TX_PTR	0x000 0x001 0x002 0x003 ... 0x0FF	<p>The transmit endpoint n pointer field contains the address of the transmit endpoint n pointer. For ISO endpoints or for non-ISO endpoints that allow double-buffering, (2*TX buffer size) must be reserved for ping-pong.</p> <p>0x000 Address = BASE 0x001 Address = BASE + 8 bytes 0x002 Address = BASE + 16 bytes 0x003 Address = BASE + 24 bytes ... 0x0FF Address = BASE + 2040 bytes</p> <p>Values after USB device controller hardware reset or USB reset are unchanged (which means that values are unknown until first write access). Pointer value must be set to a value &lt; 0xFF, because the memory size is 2K bytes and a pointer coded value = 0xFF corresponds to 2040 bytes. Addressing upper bytes results in memory overlap.</p>

**28.3.1.24 EPn\_TX[14:12].EPn\_TX\_SIZE: Transmit Endpoint n Size**

EPn\_TX.[14] bit description only applies for ISO endpoints.

This field contains the endpoint n FIFO size value. Status flags (FIFO\_EMPTY, FIFO\_FULL) and underrun condition are based on this value for all IN transactions to endpoint n.

Non-ISO:	00:	8 bytes	ISO:	000:	8 bytes
[13:12]	01:	16 bytes	[14:12]	001:	16 bytes
	10:	32 bytes		010:	32 bytes
	11:	64 bytes		011:	64 bytes
				100:	128 bytes
				101:	256 bytes
				110:	512 bytes
				111:	1023 bytes

---

**NOTE:** For ISO endpoints, a size of 1023 bytes takes up the whole memory and prevents programming with a 2K-byte USB device controller.

---

Values after USB device controller hardware reset or USB reset are unchanged (value is unknown until first write access).

**28.3.2 USB Device Transactions**

There is an interrupt to the CPU at the end of an USB transaction if the CPU has actions to perform. Isochronous transactions are an exception because isochronous interrupt information is available at start of frame interrupts. The CPU ISR code determines which endpoint and direction caused the interrupt and acts appropriately. The following subsections describe in detail the activities surrounding USB transactions that are not part of a DMA transfer. Cases where a transaction occurs before the previous one has been handled by the CPU are not taken into account in this part. The information is organized so that each section deals with one type and direction of transaction, such as:

- Non-isochronous, non-setup OUT transactions

- Non-isochronous IN transactions
- Isochronous OUT transactions
- Isochronous IN transactions

This allows each section to focus only on a specific style of transaction without adding in the confusion of special cases related to other styles.

### **28.3.3 Non-Isochronous, Non-Setup OUT (USB HOST→CPU) Transactions**

Non-isochronous, non-setup OUT transactions refer to USB transactions where data is moved from the USB host to the CPU, the USB handshaking protocols are in effect, and data transmission is ensured. These types of transactions apply to all OUT transactions on bulk and interrupt endpoint types, and to non-setup transactions on control endpoints.

[Figure 28-51](#) shows the various USB protocol conditions that can occur during non-isochronous, non-setup OUT transactions. The diagram shows the three phases that can occur in an OUT transaction, the direction of information flow for each phase, when endpoint interrupts are generated, and the resulting STAT\_FLG bits for the endpoint. The top three cases show the normal USB handshaking modes: acknowledge (good data received), NAK (device not ready to receive data), and STALL (device in a condition where the endpoint cannot handle OUT transactions). The last case is an abnormal instance where the token packet or the data packet was received with errors. The RX FIFO only contains valid receive data under the first (acknowledge) case.



**Figure 28-51. Non-Isochronous, Non-Control OUT Transaction Phases and Interrupts**

Successful data transfer to USB Host. (Endpoint STAT\_FLG.FIFO\_EN bit was set when token was received.)

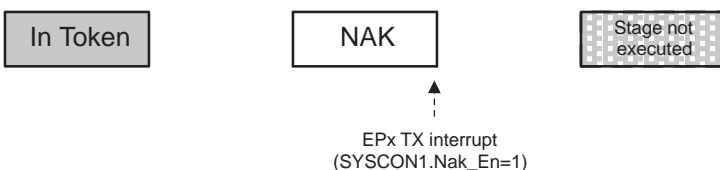


STAT\_FLG bits after interrupt

EP_Halted	STALL	NAK	ACK
0	0	0	1

After interrupt, EP's TX FIFO is empty.

No data transmitted by the LH. (Endpoint STAT\_FLG.FIFO\_EN bit was clear when token was received.)



STAT\_FLG bits after interrupt (SYSCON1.Nak\_En=1)

EP_Halted	STALL	NAK	ACK
0	0	1	0

EP TX FIFO is unchanged by this USB transaction.

EP stalled. No data transmitted by the LH. (Endpoint STAT\_FLG.EP\_HALTED bit was set when token was received or an EPO control request error has occurred.)

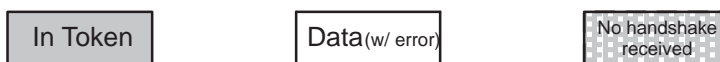


STAT\_FLG bits after interrupt

EP_Halted	STALL	NAK	ACK
1	1	0	0
or	0	1	0

EP TX FIFO is cleared by this USB transaction.

EP TX Data Error during transmission.



EP TX FIFO is unchanged by this USB transaction. No interrupt occurs. STAT\_FLG is unchanged.

- Indicates a packet received by the device
- Indicates a packet sent by the device

### 28.3.3.1 Non-Isochronous, Non-Control OUT Endpoint Handshaking Conditions

An endpoint CTRL.SET\_FIFO\_EN bit provides the main control for the endpoint ability to allow successful OUT transaction data reception for the endpoint. If at the beginning of an OUT transaction to an endpoint, the endpoint STAT\_FLG.FIFO\_EN bit is 1, the USB module is allowed to accept the OUT transaction data to the RX FIFO, and, when the transaction completes, the USB module can return ACK to the PC to indicate that the data was received correctly (this is the top case shown in Figure 28-51). If, however, the endpoint STAT\_FLG.FIFO\_EN bit was 0 at the beginning of an OUT transaction to the endpoint, the USB module returns NAK during the handshake phase to indicate that the endpoint did not accept the data (the second case shown in Figure 28-51).

The USB host need not send a whole RX FIFO worth of data to the endpoint during an OUT transaction. In this case, the RX FIFO is not full when the endpoint RX interrupt is generated. The CPU code must be careful not to read too much data. CPU code must read RXFSTAT.RXF\_COUNT value before reading data from the RX FIFO.

At the end of an USB OUT transaction to an endpoint where the data is accepted (ACKed), the hardware clears the endpoint STAT\_FLG.FIFO\_EN bit. Once the CPU software has dealt with the OUT transaction data in the endpoint RX FIFO, it must re-enable the endpoint OUT transaction reception by setting the endpoint CTRL.SET\_FIFO\_EN bit. CPU software can use the endpoint CTRL.SET\_FIFO\_EN bit as a receive flow control mechanism.

#### **28.3.3.1.1 Acknowledged Transactions (ACK)**

At completion of an OUT transaction to an endpoint, the USB module issues an endpoint-specific interrupt to the CPU and STAT\_FLG is updated. In response to the endpoint interrupt, the CPU must read EPN\_STAT register to identify the endpoint causing the interrupt, then write a 1 to the interrupt bit to clear it. The CPU must then set EP\_NUM.EP\_NUM to the endpoint number and EP\_NUM.EP\_SEL to 1, and then read the endpoint status from STAT\_FLG. STAT\_FLG.ACK is set to indicate that the endpoint received a transaction to which the USB module signaled ACK handshaking.

If STAT\_FLG.FIFO\_EMPTY is cleared, the transaction sent 1 or more bytes of data (but less than or equal to the physical size of the endpoint RX FIFO), and the data is in the endpoint RX FIFO. The CPU knows the number of bytes to read from RX FIFO by reading RXFSTAT.RXF\_COUNT value. The CPU can then read RX data from DATA register. Once the CPU has read the data from the FIFO, it sets the CTRL.SET\_FIFO\_EN bit to allow the next USB OUT transaction to the endpoint to be placed into the RX FIFO, and then clears the EP\_NUM.EP\_SEL bit. This clears the STAT\_FLG.ACK bit for this endpoint to allow next transaction status to be written into the STAT\_FLG register.

#### **28.3.3.1.2 Non-Acknowledged Transactions (NAK)**

The device can be configured via the SYSCON1.NAK\_EN either to inform the CPU of a NAKed transaction or not. If SYSCON1.NAK\_EN is cleared, no interrupt is asserted to the CPU if an OUT transaction completes with a NAK handshake and STAT\_FLG.NAK bit is not set. If SYSCON1.NAK\_EN is set, the USB module issues an endpoint-specific interrupt to the CPU at completion of an OUT transaction to an endpoint, and STAT\_FLG.NAK bit is set. In response to the endpoint interrupt, the CPU must read EPN\_STAT register to identify the endpoint causing the interrupt, and then write a 1 to the interrupt bit to clear it. The CPU must then set EP\_NUM.EP\_NUM to the endpoint number and EP\_NUM.EP\_SEL to 1, and then read the endpoint status from STAT\_FLG. STAT\_FLG.NAK is set to indicate that the endpoint received a transaction to which the USB module signaled NAK handshaking.

The CPU must set the CTRL.SET\_FIFO\_EN bit to allow the next USB OUT transaction to the endpoint to be placed into the RX FIFO, and then clear the EP\_NUM.EP\_SEL bit. This clears the STAT\_FLG.NAK bit for this endpoint to allow the next transaction status to be written into the STAT\_FLG register.

### **28.3.3.2 Non-Isochronous, Non-Control OUT Transaction Error Conditions**

#### **28.3.3.2.1 STALLed Transactions**

The USB module responds to an endpoint OUT transaction with a STALL handshake to indicate an error condition on the endpoint either if the endpoint STAT\_FLG.EP\_HALTED bit is set or if a request error occurs (control transactions only). When an endpoint OUT transaction is given a STALL handshake, the endpoint STAT\_FLG.STALL bit is set and an endpoint-specific interrupt is generated for the endpoint. STAT\_FLG.FIFO\_EN is of lower priority than STAT\_FLG.EP\_HALTED; when the STAT\_FLG.EP\_HALTED bit is set and transactions to the RX endpoint are stalled, regardless of the STAT\_FLG.FIFO\_EN value. If STAT\_FLG.FIFO\_EN is set, the STAT\_FLG.FIFO\_EN bit is automatically cleared at the end of the STALLed transaction and RX FIFO is cleared.

In response to the endpoint interrupt, the CPU must read the EPN\_STAT register to identify the endpoint causing the interrupt, and then write a 1 to the interrupt bit to clear it. The CPU must then set EP\_NUM.EP\_NUM to the endpoint number and EP\_NUM.EP\_SEL to 1, and then read the endpoint status from STAT\_FLG. STAT\_FLG.STALL is set to indicate that the endpoint received a transaction to which the USB module signaled STALL handshaking.

If `STAT_FLG.EP_HALTED` has been set by the CPU and can be removed, the CPU must set `CTRL.CLR_HALT` to clear the condition and set `CTRL.SET_FIFO_EN` to allow the next USB OUT transaction to the endpoint to be placed into the RX FIFO. If `STAT_FLG.EP_HALTED` has been set in response to a `SET_FEATURE` request sent by the USB host, or if the bit is cleared (control transaction only), the CPU has no action to perform and must clear the `EP_NUM.EP_SEL` bit. This clears the `STAT_FLG.STALL` bit for this endpoint and allows the next transaction status to be written into the `STAT_FLG` register.

#### 28.3.3.2.2 Packet Errors

In case of a receive data error during an endpoint OUT transaction (token or data packet), the USB module does not provide a handshake during the handshake phase of the transaction and no interrupt is asserted to the CPU (the fourth case shown in [Figure 28-51](#)). Additionally, the endpoint RX FIFO is not filled, and `STAT_FLG.FIFO_EN` bit is not cleared. If the CPU clears the RX FIFO during the data packet of an OUT transaction, no handshake is returned to the USB host to signal an error.

#### 28.3.3.2.3 Sequence Bit Errors

If the core does not receive expected DATA PID during an OUT transaction, the module automatically returns ACK handshake to the USB host, regardless of the `STAT_FLG.FIFO_EN` bit (per the USB specification). Data is ignored, and no interrupt is asserted to the CPU.

This error occurs if ACK handshake from previous OUT transaction is received corrupted by the USB host.

#### 28.3.3.3 Non-Isochronous, Non-Control OUT Endpoint FIFO Error Conditions

If the USB host attempts to fill more data into an endpoint RX FIFO than the FIFO can hold, a FIFO overrun occurs. The USB module does not provide a handshake during the handshake phase of the transaction and no interrupt is asserted to the CPU. Additionally, the endpoint RX FIFO is not filled, and `STAT_FLG.FIFO_EN` bit is not cleared.

The CPU must not read more data from RX FIFO than the value indicated by `RXFSTAT.RXF_COUNT`.

### 28.3.4 Non-Isochronous IN (CPU→USB HOST) Transactions

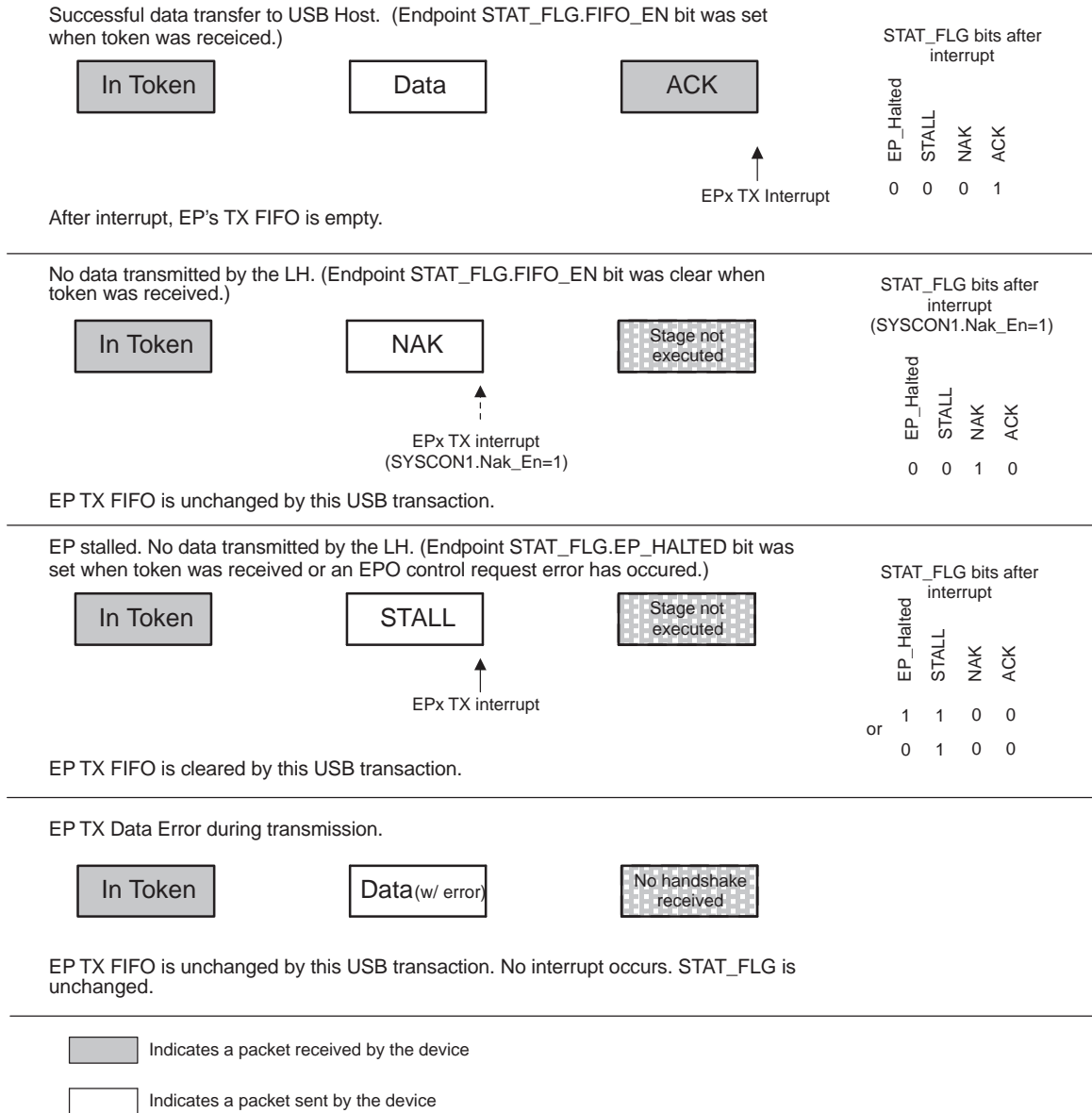
Non-isochronous IN transactions refer to USB transactions in which data is moved from the CPU to the USB host, where the USB handshaking protocols are in effect, and data transmission is ensured. These transactions are the IN transactions that occur on control, bulk, and interrupt endpoints. These transactions do not ensure USB bandwidth.

To provide data for an endpoint IN transaction, the CPU code writes the transmit data into the endpoint transmit FIFO. CPU code must first wait until the USB is done with any previous TX data for the endpoint (if data had previously been written to the TX FIFO). This must be done by proper response to endpoint-specific transmit interrupts. When an IN transaction to the endpoint occurs, if the endpoint `STAT_FLG.FIFO_EN` bit is set, the USB module sends any data that is in the endpoint TX FIFO during the data phase. If the TX FIFO is empty and `STAT_FLG.FIFO_EN` is set when an IN transaction to the endpoint occurs, a 0-byte data packet is sent.

Once the endpoint previous transmit activity is taken care of, the CPU code gains access to endpoint FIFO and status by setting the `EP_NUM.EP_SEL` bit. Then the CPU can write the new transmit data to the endpoint TX FIFO via the DATA register (being careful not to overflow the FIFO). Once all of the transmit data has been written to the endpoint FIFO, CPU code sets the `CTRL.SET_FIFO_EN` bit to allow the USB to use the endpoint TX FIFO, and then clears the `EP_NUM.EP_SEL` bit. The data in the endpoint TX FIFO is sent to the USB host the next time an IN transaction to the endpoint occurs.

[Figure 28-52](#) shows the various USB protocol conditions that can occur during non-isochronous IN transactions. It diagrams the three phases of the IN transaction, the direction of information flow for each phase, when endpoint-specific interrupts are generated, and the resulting `STAT_FLG` bits for the endpoint. The top three cases show the normal USB handshaking: acknowledge (data sent by USB module and received properly by the USB host), NAK (device not ready to send data to USB host), and STALL (device in a condition where the endpoint cannot handle IN transactions). The last case shows an abnormal case in which there is an error either in the token packet received by the core or in the data packet received by the USB host.

**Figure 28-52. Non-Isochronous IN Transaction Phases and Interrupts**



### 28.3.4.1 Non-Isochronous IN Endpoint Handshaking

Per USB specifications for IN transactions, the USB host can provide only one of two handshakes to the USB function during the handshake phase: ACK or no handshake at all. The first indicates successful transfer (first case shown in [Figure 28-52](#)), and the second indicates that the host received a garbled data packet (last case shown in [Figure 28-52](#)).

#### 28.3.4.1.1 Acknowledged Transactions (ACK)

When the endpoint IN transaction completes on the USB bus with an ACK handshake, the endpoint generates an endpoint-specific interrupt to the CPU (see first case in [Figure 28-52](#)). In response to the endpoint interrupt, the CPU must read the EPN\_STAT register to identify the endpoint causing the interrupt, and then write a 1 to the interrupt bit to clear it. The CPU must then set EP\_NUM.EP\_NUM to the endpoint number, EP\_NUM.EP\_DIR to 1 (to signal an IN endpoint), and EP\_NUM.EP\_SEL to 1, and then read the endpoint status from STAT\_FLG. STAT\_FLG.ACK is set to indicate that the endpoint received an ACK handshake from the USB host, and the TX FIFO is empty (because any data that was in the TX FIFO was transmitted during the IN transaction).

If the CPU has more data to transmit to the USB host, it must fill the TX FIFO following the process indicated above. It must then clear EP\_NUM.EP\_SEL bit. This clears the STAT\_FLG.ACK bit for this endpoint to allow next transaction status to be written into the STAT\_FLG register.

### 28.3.4.1.2 Non-Acknowledged Transactions (NAK)

For the case in which the CPU is not ready to provide transmit data for transactions to an IN endpoint, the core provides a NAK handshake to the host for any USB IN transaction to that endpoint. Readiness to transmit data is signaled via the endpoint STAT\_FLG.FIFO\_EN bit; when 1 it indicates that data in the TX FIFO can be sent to the USB host. When the endpoint STAT\_FLG.FIFO\_EN bit is 0 and an IN transaction to the endpoint occurs, NAK handshake is sent, indicating that the CPU is not ready to handle the request.

If SYSCON1.NAK\_EN bit is cleared, when the NAK handshake is sent in the data packet portion of the transaction to the IN endpoint, STAT\_FLG is not updated and no endpoint-specific interrupt to the CPU is generated. If the SYSCON1.NAK\_EN bit is set, when the NAK handshake is sent in the data packet portion of the transaction to the IN endpoint, the STAT\_FLG.NAK bit is set and an endpoint-specific interrupt to the CPU is generated.

In response to the endpoint interrupt, the CPU must read the EPN\_STAT register to identify the endpoint causing the interrupt, and then write a 1 to the interrupt bit to clear it. The CPU must then set EP\_NUM.EP\_NUM to the endpoint number, EP\_NUM.EP\_DIR to 1 (to signal an IN endpoint), and EP\_NUM.EP\_SEL to 1, and then read the endpoint status from STAT\_FLG. STAT\_FLG.NAK is set to indicate that the endpoint sent a NAK handshake to the USB host. If the CPU has data to transmit to the USB host, it must fill the TX FIFO following the process indicated above. The CPU must then clear the EP\_NUM.EP\_SEL bit. That clears the STAT\_FLG.NAK bit for this endpoint to allow the next transaction status to be written into the STAT\_FLG register. Signaling NAK does not cause the endpoint TX FIFO to be cleared (because the CPU still retains control of the FIFO).

Signaling NAK handshake for several endpoint transactions in a row can cause the PC host to discard the transaction, so NAK is not necessarily a good mechanism in cases where the CPU is not able to service a request for long periods of time.

### 28.3.4.2 Non-Isochronous IN Transaction Error Conditions

#### 28.3.4.2.1 STALLed Transactions

The USB module sends a STALL handshake to the USB host during the data phase of the transaction to the IN endpoint either if the endpoint STAT\_FLG.EP\_HALTED flag is set, or if a request error occurs (control transaction only). A USB STALL handshake indicates that the device endpoint is in a condition in which it is not able to transfer data and instructs the USB host not to retry the transaction. The device typically requires intervention via some other mechanism to clear the condition, usually a control transfer via endpoint 0. The CPU can set the endpoint EP\_HALTED bit by selecting the endpoint by writing the appropriate value in EP\_NUM register, and then setting the endpoint CTRL.Set\_HALT bit, and clear it by selecting the endpoint, and then setting the endpoint CTRL.Clr\_HALT bit. When the endpoint EP\_HALTED bit is set, the endpoint signals STALL for its IN transactions until the HALT condition is cleared. When the STALL handshake is sent in response to a transaction to the endpoint, the STAT\_FLG.STALL bit is set, and an endpoint-specific interrupt to the CPU is generated.

In response to the endpoint interrupt, the CPU must read the EPN\_STAT register to identify the endpoint causing the interrupt, and then write a 1 to the interrupt bit to clear it. The CPU must then set EP\_NUM.EP\_NUM to the endpoint number, EP\_NUM.EP\_DIR to 1 (to signal an IN endpoint), and EP\_NUM.EP\_SEL to 1, and then read the endpoint status from STAT\_FLG. STAT\_FLG.STALL is set to indicate that the endpoint sent a STALL handshake to the USB host. The CPU must then clear EP\_NUM.EP\_SEL bit. This clears the STAT\_FLG.STALL bit for this endpoint and allows the next transaction status to be written into the STAT\_FLG register.

Except for control endpoint 0, separate endpoint halt bits are defined for each direction; so for a given endpoint number, the TX can be halted when the RX is not.

### **28.3.4.2.2 Packet Errors**

If an error (CRC, bit stuffing, or PID check) occurs during the token packet of a USB IN transaction to a non-isochronous endpoint, the USB block ignores the transaction. No endpoint-specific interrupt to the CPU occurs for transactions with corrupted packets. If the CPU clears the TX FIFO during the data packet of an IN transaction, a bit stuffing error is forced.

If the USB host returns no handshake after an IN transaction (in case of an error during transmission), the USB device controller module detects after a time-out that an error has occurred. The data to transmit is still in the TX FIFO to be re-sent during next IN transaction, STAT\_FLG.FIFO\_EN is not cleared, and no interrupt is asserted to the CPU.

### **28.3.4.3 Non-Isochronous IN Endpoint FIFO Error Conditions**

The CPU cannot write more data into the TX FIFO than the configured FIFO size.

### **28.3.5 Isochronous OUT (USB HOST→CPU) Transactions**

Isochronous OUT transactions are USB transactions in which a given amount of data is transferred from the USB host to the USB device controller module device every 1-ms USB frame. No USB handshaking is provided, and no endpoint-specific interrupt to the CPU is generated at completion of an isochronous OUT transaction. The CPU is responsible for handling isochronous OUT data at each start of frame (SOF) interrupt.

At every SOF interrupt, for each isochronous OUT endpoint, CPU code must select the endpoint by writing the appropriate value in the EP\_NUM register and check STAT\_FLG.ISO\_FIFO\_EMPTY. If the RX FIFO contains data, code must read the RXFSTAT.RXF\_COUNT value (if the number of bytes to read from RX FIFO is not known), read all the bytes from RX FIFO via the data register, and then clear the EP\_NUM.EP\_SEL bit.

Because the USB transaction for the isochronous endpoint can occur at any time during the USB 1-ms frame, the USB interface implements a double-buffering of the endpoint receive data FIFO. The endpoint includes two FIFOs, each of which is the length of the configured isochronous endpoint. At all times, one of the two FIFOs is foreground and the other is background. The USB interface side of the USB module is allowed to write to the background RX FIFO, and the CPU is allowed to read to the foreground RX FIFO. The designations foreground and background are swapped at each start of frame (SOF). Isochronous endpoint FIFOs in the background are always enabled to the USB, whereas the foreground FIFOs are enabled to the CPU.

[Figure 28-53](#) shows the two phases (ISO OUT token and data) of an isochronous OUT data transfer in the top portion of the figure. No endpoint-specific interrupt to the CPU is generated for the isochronous OUT transaction. The data for isochronous endpoints are instead handled by the CPU at each start of frame (SOF) interrupt, which is shown as the second case in [Figure 28-53](#).



**Figure 28-53. Isochronous OUT Transaction Phases and Interrupts**

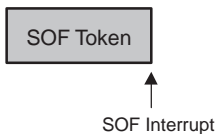
Successful data transfer from USB Host



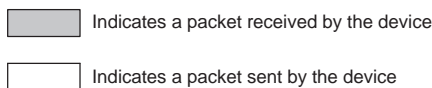
No handshake occurs. EP RX FIFO contains received data after data packet completes. No interrupt occurs.

Reception of SOF causes SOF interrupt.

Note: An SOF interrupt is generated even if the SOF packet is corrupted.



LH code for SOF interrupt service routine must fill all isochronous in EP TX FIFOs with new transmit data and pull new receive data from all Isochronous Out EP RX FIFOs.



### 28.3.5.1 Isochronous OUT Endpoint Handshaking

Because isochronous endpoint transactions have no handshake packets, the STAT\_FLG.STALL, STAT\_FLG.NAK, and STAT\_FLG.ACK bits for isochronous endpoints always return 0. Because there is no handshake, the endpoint-specific interrupt for isochronous endpoints is not used.

### 28.3.5.2 Isochronous OUT Transaction Error Conditions

If the CPU fails to read all of the data in the ISO OUT endpoint foreground FIFO by the time the foreground and background FIFOs are switched (at the next SOF), the endpoint FIFO that is being switched to the background is flushed, and the STAT\_FLG.DATA\_FLUSH bit is asserted for the duration of the next frame.

There is no special indication for the case in which the USB host does not provide a transaction to an ISO OUT endpoint during a frame, but once the FIFO that was background in that frame is foreground, the FIFO is empty (a 0-length data ISO OUT transaction also results in an empty FIFO and cannot be distinguished from a missed ISO OUT transaction).

If an ISO OUT transaction occurs with data error (CRC, PID check, or bit stuffing), the RX FIFO is empty at the next SOF interrupt, and the STAT\_FLG.ISO\_ERR bit is asserted for the duration of the next frame.

### 28.3.5.3 Isochronous OUT Endpoint FIFO Error Conditions

The CPU must never read more data than the value given by RXFSTAT.RXF\_COUNT.

If the USB host sends more data than the FIFO can contain, the FIFO is cleared and the STAT\_FLG.ISO\_ERR is set at the next SOF interrupt. A properly configured USB system does not do this.

**NOTE:** Both foreground and background isochronous FIFOs are cleared when the CTRL.CLR\_EP bit is set.

### 28.3.6 Isochronous IN (CPU→USB HOST) Transactions

Isochronous IN transactions are USB transactions in which a given amount of data is transferred from the USB device controller module device to the USB host every 1-ms USB frame. No handshaking is provided.

The USB module provides double-buffering of data for ISO IN endpoints; the background FIFO is used as the source of data for IN transactions to the ISO endpoint, and the foreground FIFO can be written to by the CPU. When an IN transaction to an ISO endpoint occurs, the USB module sends all data found in the endpoint background TX FIFO. The CPU is responsible for providing new data to the isochronous IN endpoint foreground TX FIFO at each start of frame interrupt.

In response to the SOF interrupt, for each isochronous IN endpoint, CPU code selects the endpoint (via the EP\_NUM register), and then fills the endpoint TX FIFO (via the DATA register). Once all the transmit data have been written to the FIFO, the CPU code must clear the EP\_NUM.EP\_SEL bit.

Because the USB transaction for the isochronous endpoint can occur at any time during the USB 1-ms frame, the USB interface implements a double-buffering of the endpoint transmit data FIFO. The endpoint includes two FIFOs, each of which is the length of the configured isochronous endpoint. At all times, one of the two FIFOs is foreground and the other is background. The USB interface side of the USB module is allowed to read from the background TX FIFO, and the CPU is allowed to write to the foreground TX FIFO. The designations foreground and background are swapped, and the new background TX FIFO is cleared at each start of frame (SOF). Because ISO endpoints implement double-buffering, ISO endpoints do not control access to the FIFOs via a CTRL.SET\_FIFO\_EN bit; the CTRL.SET\_FIFO\_EN and the STAT\_FLG.FIFO\_EN bits are not implemented for ISO IN endpoints.

Figure 28-54 shows the transaction phases associated with isochronous IN transactions and the SOF transaction. No endpoint-specific interrupt to the CPU is generated as a result of an isochronous IN transaction, and there is no handshake phase. The SOF transaction causes an SOF interrupt to the CPU; it is assumed that the CPU refills the isochronous IN endpoint transmit FIFO at each SOF interrupt.

**Figure 28-54. Isochronous IN Transaction Phases and Interrupts**

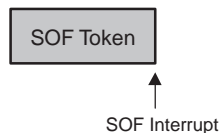
Successful data transfer to PC host



No handshake occurs. EP RX FIFO is empty after data sent. No EP interrupt occurs. STAT\_FLG is unchanged.

Reception of SOF causes SOF interrupt.

Note: An SOF interrupt is generated even if the SOF packet is corrupted.



LH code for SOF ISR must fill all isochronous In EP TX FIFOs with new transmit data and pull new receive data from all isochronous Out EP RX FIFOs.

- Indicates a packet received by the device
- Indicates a packet sent by the device



### 28.3.6.1 Isochronous IN Endpoint Handshaking

Because isochronous endpoint transactions have no handshake packets, the STAT\_FLG.STALL, STAT\_FLG.NAK, and STAT\_FLG.ACK bits for isochronous endpoints always return 0. Because there is no handshake, there is no endpoint-specific interrupt to the CPU to report handshake results for isochronous endpoints.

### 28.3.6.2 Isochronous IN Transaction Error Conditions

If the USB host did not successfully complete an ISO IN transaction in the previous frame, and if data were present in TX FIFO to be sent at the IN transaction, the STAT\_FLG.MISS\_IN bit is asserted for the duration of the following frame. If the ISO IN endpoint is cleared in the middle of a USB transaction to the background FIFO, the macro forces a bit stuffing error for the ISO transaction.

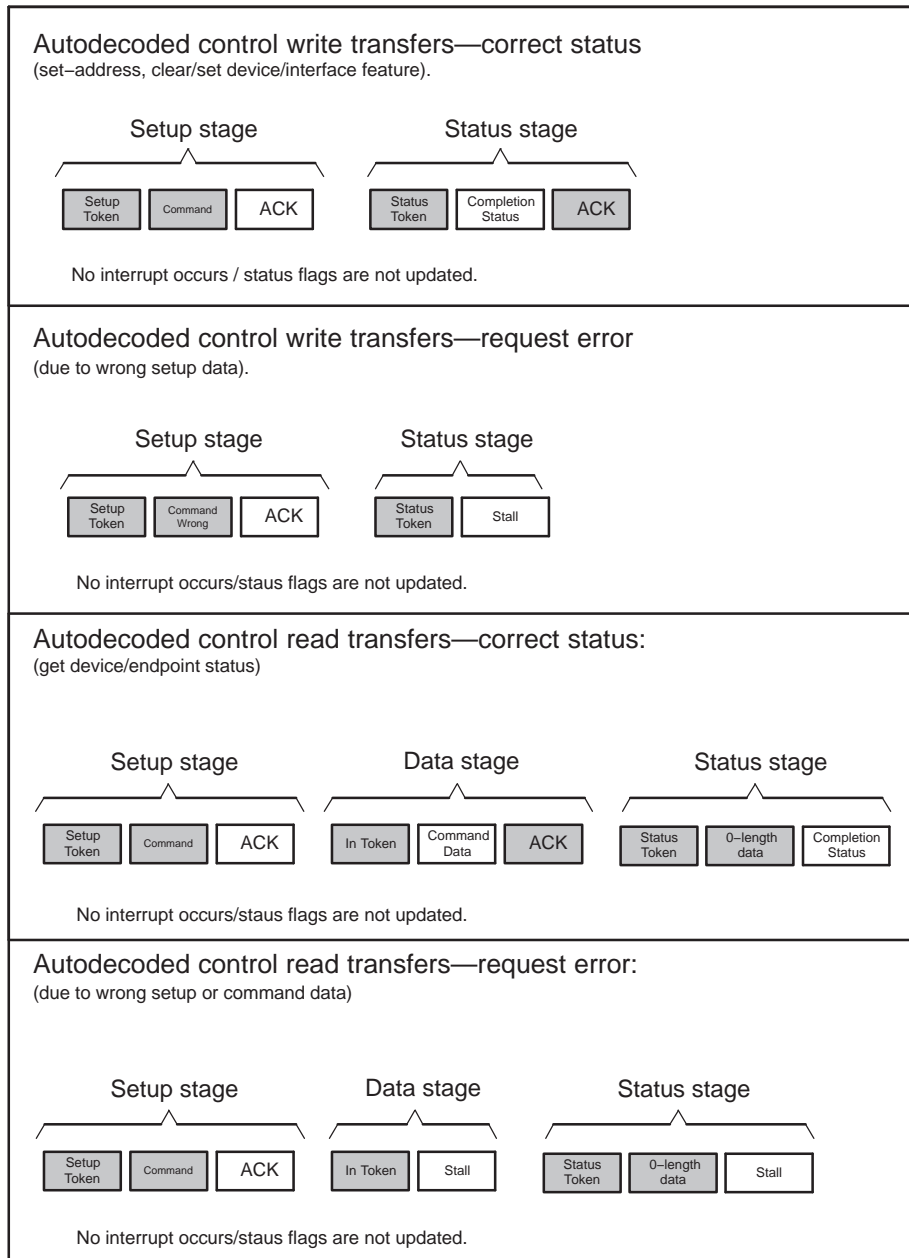
### 28.3.6.3 Isochronous IN Endpoint FIFO Error Conditions

If the CPU attempts to overfill the configured endpoint FIFO, data written to DATA register after the TX FIFO is full is lost, but any data that was successfully put into the FIFO is transmitted when that FIFO is the background FIFO and an IN transaction for that endpoint occurs. Because an ISO TX FIFO is cleared automatically on the toggle from background to foreground, there is no reason to clear the FIFO. However, if the CPU does not wish to send the data it wrote, clearing the endpoint is the only mechanism to do this.

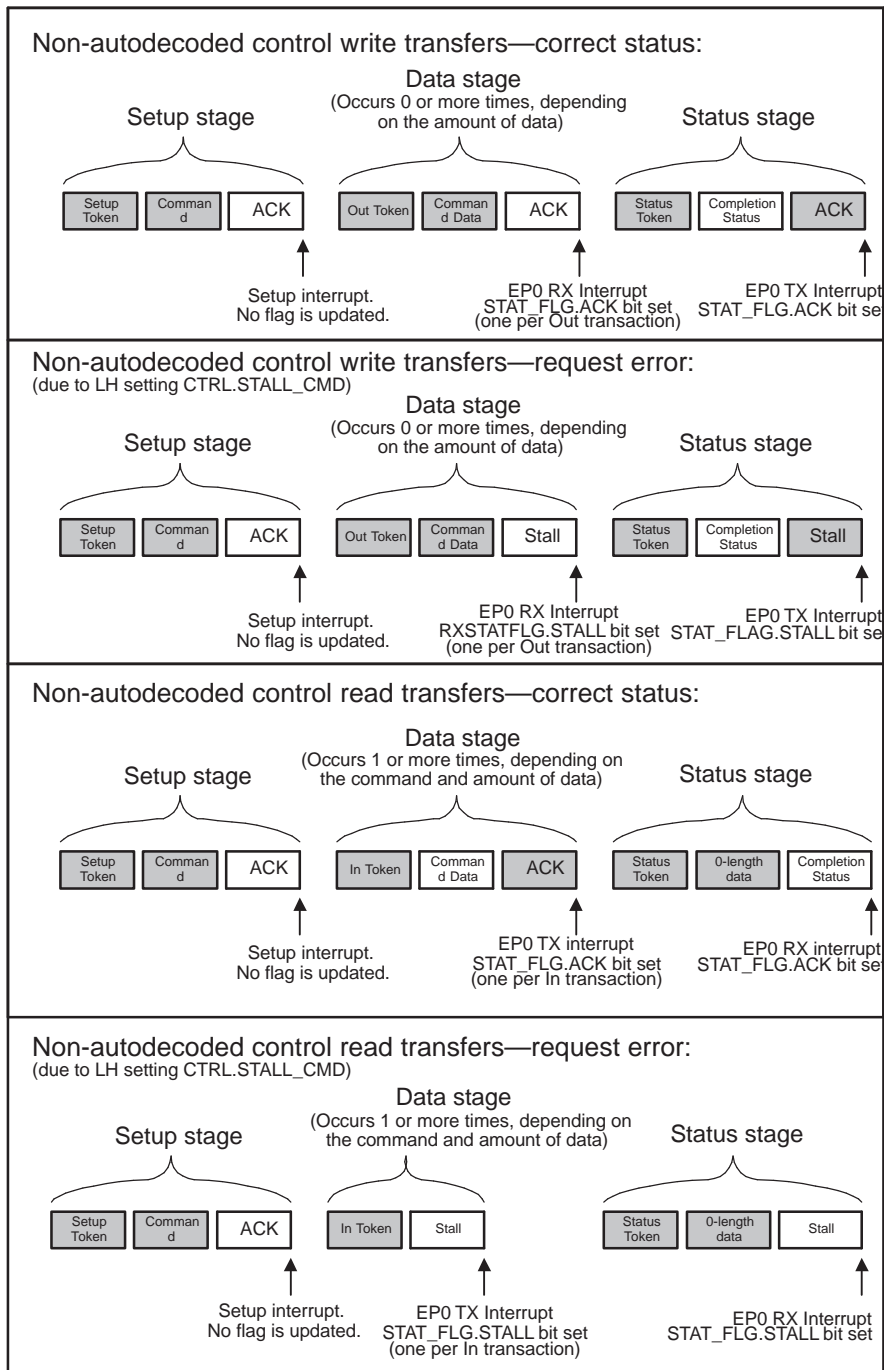
## 28.3.7 Control Transfers on Endpoint 0

Control transfers on endpoint 0 include control write and control read transfers. Control write and control read transfers are each composed of two or more transactions to endpoint 0. Additionally, the USB device controller module is capable of autodecoding some control write and control read transfers. These operations are summarized in [Figure 28-55](#) and [Figure 28-56](#). An IN or an OUT transaction is received out of a control request. This transaction is automatically stalled by the core.

**Figure 28-55. Stages and Transaction Phases of Autodecoded Control Transfers**



**Figure 28-56. Stages and Transaction Phases of Non-Autodecoded Control Transfers**



Non-autodecoded control read and control write transfers are sets of transactions that occur on endpoint 0 and have specific USB protocol meaning but are not handled automatically by the core. The USB device controller block automatically provides an ACK handshake for the setup stage transaction, but the data and status stage transaction handshaking is accomplished using the usual RX and TX control bits that affect transaction handshaking. A general USB interrupt to the CPU occurs at the end of each transaction of each stage of a control transfer. The CPU must perform the following actions to act on non-autodecoded control transfers:

- Process the setup phase setup USB interrupt. The CPU reads the control transfer command from the setup FIFO and decodes the command. For control reads, the CPU fetches the requested read data and places it (or as much of the read data as fits) into the endpoint 0 FIFO, and then enables the endpoint 0 FIFO. For control writes, the CPU code only enables the endpoint 0 FIFO. CPU code also sets any flags needed for processing endpoint 0 USB interrupts during the control transfer.
- Process the data phase endpoint 0 general USB interrupt(s). For control reads, the data phase general USB endpoint 0 TX interrupt indicates that the previously provided transmit data has been sent. Any additional data must be written to the endpoint 0 FIFO. For control writes, the write data must be pulled from the endpoint 0 FIFO, and when all control write data is available, interpret the write data and act on the write request. After handling the last data phase interrupt, the CPU must set the endpoint 0 control bits to signal the desired status to the host.
- Process the status stage endpoint 0 general USB interrupt. The CPU provides its completion status back to the USB host during this stage, either via status in the data phase of the transaction (for control write transfers) or via the handshake phase of the transaction (for control read transfers).

Autodecoded control read and control write transfers are sets of transactions that occur on endpoint 0 that have specific USB protocol meaning and are handled automatically by the USB device controller block without any intervention by the CPU. The USB device controller block handles all handshaking automatically and without regard to the endpoint 0 control bits that affect normal (non-control transfer) transaction handshaking. No interrupt is asserted to the CPU during autodecoded control transfers.

If no USB 1.1 specification defined request is associated with the data of the setup phase, request is stalled by the core and the CPU is not informed of its occurrence (autodecoded).

When a setup token is identified, the USB decode module must monitor the setup stage data packet, decode it, and determine whether it is an autodecoded or a non-autodecoded transfer and a control read or a control write. If it is a valid non-autodecoded request, the setup FIFO is immediately cleared and control of the FIFO is immediately taken away from the CPU (if the CPU had control of the FIFO). New setup data are placed into the setup FIFO, and the setup interrupt flag is set (IRQ\_SRC.setup).

In response to the setup interrupt, the CPU must select the setup FIFO by setting the EP\_NUM.SETUP\_SEL bit. This clears the IRQ\_SRC.SETUP flag. The CPU must then read 8 bytes from the setup FIFO, clear the EP\_NUM.SETUP\_SEL bit, and confirm that IRQ\_SRC.SETUP bit has not been reset by a new setup transaction. If the IRQ\_SRC.SETUP flag is asserted, the CPU must discard the previously read data and handle the new setup packet as explained above. Thus the CPU never misses a new occurring setup transaction (per USB 1.1 specification).

### **28.3.7.1 Autodecoded Control Write Transfers**

For set address control write transfers, the USB address provided in the setup token is captured to the USB module device address register. If new address is different from 0, the device moves into addressed state (DEVSTAT.ADD set) if it was not already addressed.

For set and clear feature control writes, the appropriate feature information bit is set or cleared. When a set or clear feature transfer occurs to set or clear the device remote wake-up feature, the DEVSTAT.R\_WK\_OK bit is set or cleared, as appropriate. If a set or clear interface feature occurs, the request is automatically stalled by the core because no feature is defined for interface (see the USB 1.1 specification).

Per the USB 1.1 specifications, a SET\_ADDRESS request is effective after the status stage of the request, even if the status stage does not end with an ACK handshake. SET/CLEAR\_FEATURE requests are effective after setup stage, even if no status stage occurs.

#### **28.3.7.1.1 Autodecoded Control Write Transfer Handshaking**

The USB device controller module automatically provides ACK handshaking for all transactions of all stages of autodecoded control write transfers, except if a corrupted packet is received, which the USB module ignores. The CTRL.SET\_FIFO\_EN and SYSCON2.STALL\_CMD bits have no effect on handshaking.

### 28.3.7.1.2 Autodecoded Control Write Transfer Error Conditions

If the token packet or the data packet of a setup stage transaction has an error (bad CRC, PID check, or bit stuffing error), the USB block ignores the transaction. The USB block does not provide ACK handshaking in this case.

### 28.3.7.2 Autodecoded Control Read Transfers

Autodecoded control reads include the standard device requests GET\_ENDPOINT and DEVICE\_STATUS. These control read transfers access information that is kept in registers inside the USB module, so CPU code is not involved in filling the read data into the TX FIFO.

The USB module returns the currently selected appropriate status information (depending on the wIndex value in the setup stage data packet) during the data phase of the single IN transaction of the data stage, and provides ACK as the handshake for the status stage handshake phase. The CPU receives no interrupt.

#### 28.3.7.2.1 Autodecoded Control Read Transfer Handshaking

The USB device controller module automatically provides ACK handshaking for all transactions of all stages of autodecoded control read transfers, except if a corrupted token packet is received, which the USB module ignores. The CTRL.FIFO\_EN and SYSCON2.STALL\_CMD bits have no effect on the handshaking. If the status packet has a DATA0 PID instead of a DATA1 PID, status is STALLED and no interrupt is asserted to the CPU. If the setup packet has a DATA1 PID instead of a DATA0 PID, setup transaction is ignored (error).

#### 28.3.7.2.2 Autodecoded Control Read Transfer Error Conditions

If the token phase or the data phase of a setup stage transaction has an error (bad CRC, PID check, or bit stuffing error), the USB block ignores the transaction. The USB block does not provide ACK handshaking in this case.

Data errors during the data stage of autodecoded control write transfers are handled in the standard way; any data stage transaction from the host where a data error occurs is ignored.

It is possible that the USB host sends a GET\_ENDPOINT/DEVICE\_STATUS request with a bad parameter. If the autodecode mechanism senses a bad parameter in the setup stage data phase, the autodecode mechanism causes a STALL handshake to be signaled during the data phase of the data stage and during the status stage.

### 28.3.7.3 Non-Autodecoded Control Write Transfers

Non-autodecoded control write transfers include the SET\_/CLEAR\_ENDPOINT feature, SET\_CONFIGURATION, SET\_INTERFACE, SET\_DESCRIPTOR and class- or vendor-specific control write transfers. Non-autodecoded control write transfers consist of two or three stages [setup, data (optional), and status].

The setup stage of a valid non-autodecoded control write transfer consists of one SETUP transaction from USB host to USB device. At the end of the setup stage handshake, the USB module generates a CPU general USB interrupt with the IRQ\_SRC.SETUP flag set. The CPU must respond to this general USB interrupt by setting EP\_NUM.SETUP\_SEL bit, which clears the setup interrupt flag. The CPU must then read 8 bytes from the setup FIFO via the DATA register, clear EP\_NUM.EP\_SEL bit, and check the IRQ\_SRC.SETUP flag. If the IRQ\_SRC.SETUP flag is set, the CPU must discard the setup data it has just read and handle the new setup data packet following the same scheme. If the IRQ\_SRC.SETUP flag is cleared, the CPU code interprets this request information and performs any application-specific activity needed because of the setup stage request. If there is one or more data stage for the transfer, the CPU must set the CTRL.SET\_FIFO\_EN bit for endpoint 0 to allow the core to accept RX data from the coming OUT transaction.

The data stage for non-autodecoded control writes consists of zero or more OUT transactions. Transaction handshaking and interrupt generation as for non-isochronous, non-control OUT endpoints applies. The CPU can cause NAK, STALL, or ACK signaling for the data stage transactions. If ACK was signaled on a given general USB interrupt, the CPU must respond by reading the data from the endpoint 0 RX FIFO and saving it for processing.

After completion of the data stage, a status stage IN transaction occurs. The USB module provides handshaking to the USB host based on the endpoint 0 handshaking control bit `STAT_FLG.FIFO_EN`. The CPU can delay signaling completion of the control write transfer by forcing NAK handshaking to the host during the status stage (by holding `STAT_FLG.FIFO_EN` 0), or causing ACK handshaking by setting the `CTRL.SET_FIFO_EN` bit with an empty endpoint 0 FIFO. An endpoint 0 TX general USB interrupt is sent to the CPU at completion of the status stage.

After a `SET_CONFIGURATION` request, the device moves in addressed or configured state as soon as the CPU sets the `SYSCON2.DEV_CFG` or `SYSCON2.CLR_CFG` bits.

### **28.3.7.3.1 Specific CPU Required Actions**

If the device receives a valid set endpoint halt feature request, it must set the appropriate `CTRL.SET_HALT` control bit.

If the device receives a valid `CLEAR_ENDPOINT` halt feature request, it must set the appropriate `CTRL.RESET_EP` bit to clear the halt condition, FIFO flags, and reset data PID to `DATA0` for the endpoint. If the specified endpoint number is 0, the CPU has only to set `CTRL.CLR_HALT` bit to clear the halt condition.

If the device receives a valid `SET_CONFIGURATION` request, it must reset all endpoints by setting the `CTRL.RESET_EP` control bits, set the `SYSCON1.SELF_PWR` bit to the appropriate value, and then set halt conditions for endpoints not used by the default interface set for the configuration. If the device was addressed when the `SET_CONFIGURATION` was received, the CPU must write 1 to the `SYSCON2.DEV_CFG` bit to allow the device to move into the configured state (`DEVSTAT.CFG` bit set). If the device was configured when the `SET_CONFIGURATION` was received, and the new configuration value is 0, the CPU must write 1 to the `SYSCON2.CLR_CFG` bit to allow the device to move back into the addressed state (`DEVSTAT.CFG` bit cleared).

If the device receives a valid set interface request, it must reset all endpoints used by the interface set, by setting `CTRL.RESET_EP` control bits, and then set halt conditions for endpoints not used by this interface.

Other CPU required actions are specific to the request and not detailed in this document.

### **28.3.7.3.2 Non-Autodecoded Control Write Transfer Handshaking**

Setup stage transactions that are valid are signaled ACK. Transactions with invalid setup stage token or data packets are ignored and receive no handshake packet from the USB module, and there is no interrupt generated.

Data stage handshaking for non-autodecoded control write transfers is dependent on the endpoint 0 `STAT_FLG.FIFO_EN`, `STAT_FLG.EP_HALTED`, and `SYSCON2.STALL_CMD` bits. The CPU can delay completion of any transaction of the data stage by signaling NAK (via `CTRL.SET_FIFO_EN` bit not set). The USB specification requires that once STALL is signaled in a control transfer, it must be signaled on that endpoint until the next setup token is received. Either the `SYSCON2.STALL_CMD` or the `CTRL.SET_HALT` (reflected in the `STAT_FLG.EP_HALTED` register bit) register bits provide this functionality. `STAT_FLG.EP_HALTED` does not reflect the forced STALL caused by `SYSCON2.STALL_CMD`; it retains its previous value.

Status stage handshaking is controlled by the endpoint 0 `STAT_FLG.FIFO_EN` and `SYSCON2.STALL_CMD` bits. Successful completion of a non-autodecoded control write transfer is indicated by the USB device controller module returning a zero length data payload for the data phase of the status stage and an ACK handshake from the host for the handshake phase of the status stage. Although NAK handshaking can be used to indicate delays in completion of the requested control write, the USB host can choose to abort the control write after some number of NAKs.



### 28.3.7.3.3 Non-Autodecoded Control Write Transfer Error Conditions

If an error occurs while dealing with the control write, which the CPU cannot deal with itself, it must signal STALL to the USB host for all subsequent transactions until a new setup token to endpoint 0 occurs. This is true for both data stage and status stage transactions. This is most conveniently done by setting endpoint 0 SYSCON2.STALL\_CMD bit, which causes stalling of all the remaining transactions of all remaining stages of a non-autodecoded control transfer, up to the reception of the next valid SETUP command.

Error conditions are handled as for BULK/INTERRUPT transactions. If a packet is received corrupted, the core ignores the transaction and no interrupt is asserted.

### 28.3.7.4 Non-Autodecoded Control Read Transfers

Non-autodecoded control read transfers include the GET\_INTERFACE\_STATUS, GET\_CONFIGURATION, GET\_INTERFACE, GET\_DESCRIPTOR, SYNCH\_FRAME and class- or vendor-specific control read transfers. Non-autodecoded control read transfers consist of three stages (setup, data, and status).

The setup stage of a valid non-autodecoded control read transfer consists of one SETUP transaction from USB host to USB device. At the end of the setup stage handshake, the USB module generates a CPU general USB interrupt with the IRQ\_SRC.SETUP flag set. The CPU must respond to this general USB interrupt by setting the EP\_NUM.SETUP\_SEL bit, which clears the setup interrupt flag. The CPU must then read 8 bytes from the setup FIFO via the DATA register, clear the EP\_NUM.EP\_SEL bit, and check the IRQ\_SRC.SETUP flag. If the IRQ\_SRC.SETUP flag is set, the CPU must discard the setup data it has just read and handle the new setup data packet following the same scheme. If the IRQ\_SRC.SETUP flag is cleared, the CPU code interprets this request information and then prepares data for the IN transaction that follow. This includes placing the data being requested (or the first few bytes, if more than one FIFO worth of data is being returned) into the endpoint 0 FIFO, and setting the CTRL.SET\_FIFO\_EN bit.

The data stage of a control read transfer consists of one or more IN transactions. Transaction handshaking and interrupt generation as for non-isochronous, non-control IN endpoints applies; the CPU can cause NAK, STALL, or ACK signaling for the data stage transactions. At endpoint 0 TX general USB interrupts, CPU code must move more data to the endpoint 0 FIFO, until the last bytes of the requested data have been provided. Although SETUP packets have a defined payload length, the USB host can cancel the transaction at any time, without the status stage, and resend another SETUP command. The CPU code must be able to operate correctly in this situation.

After completion of the data stage, a status stage OUT transaction occurs. The USB host sends a zero-length data packet, and the CPU code must return its completion status for the control read standard request via standard handshaking mechanisms.

---

**NOTE:** In the case of returning exactly what the host requested and that request was a multiple of the maximum packet size, no zero-length packet is required. A zero-length packet is required only when the amount of data the device has to return is less than the amount requested by the host and the amount returned is a multiple of the maximum packet size (source USB forum).

---

#### 28.3.7.4.1 Non-Autodecoded Control Read Transfer Handshaking

Handshaking for the setup stage of non-autodecoded control read transfers is forced by the USB module to always be ACK, unless there is a data error in the packet, in which case the USB module ignores the transaction. If the setup packet has a DATA1 PID instead of a DATA0 PID, the setup transaction is ignored (error).

Data stage handshaking for non-autodecoded control read transfers is dependent on the endpoint 0 STAT\_FLG.FIFO\_EN, STAT\_FLG.EP\_HALTED, and SYSCON2.STALL\_CMD bits. The handshaking information is used during the data phase of the data stage transaction. The USB specification requires that once STALL is signaled in a control transfer, it must be signaled until the next setup token is received. The SYSCON2.STALL\_CMD and CTRL.SET\_HALT (reflected through the STAT\_FLG.EP\_HALTED register bit) register bits provide this functionality. STAT\_FLG.EP\_HALTED does not reflect the forced STALL caused by SYSCON2.STALL\_CMD; it retains its previous value.

The status stage is controlled by the CTRL.FIFO\_EN and SYSCON2.STALL\_CMD bits.

Successful completion of non-autodecoded control read transfers is indicated by the host sending an OUT token followed by an empty packet and the USB device controller responding with ACK. If the data packet sent by the USB host during the status stage of a control read request is not empty, the OUT transaction is accepted by the core, but OUT data is not put into the endpoint 0 RX FIFO. If the status packet has a DATA0 PID instead of a DATA1 PID, a STALLED is returned by the core and an interrupt is asserted.

#### 28.3.7.4.2 Non-Autodecoded Control Read Transfer Error Conditions

If an error occurs while dealing with the control read, which the CPU cannot deal with itself, it must signal STALL to the USB host for all subsequent transactions until a new setup token to endpoint 0 occurs. This is true for both data stage and status stage transactions. This is most conveniently done by setting endpoint 0 SYSCON2.STALL\_CMD bit, which causes stalling of all the remaining transactions of all remaining stages of a non-autodecoded control transfer, up to the reception of the next valid SETUP command.

Error conditions are handled as for BULK/INTERRUPT transactions. The USB device controller module responds to control read status stage transactions that have a bad token or bad data by not sending a handshake packet. In both cases, the transaction is ignored and no general USB interrupt is generated to the CPU.

#### 28.3.7.5 Autodecoded Versus Non-Autodecoded Control Requests

**Table 28-55. Autodecoded Versus Non-Autodecoded Control Requests** <sup>(1) (2) (3) (4)</sup>

Request	Recipient	Status	CPU Required Action	Device Behavior if Device Is Not Configured
GET_STATUS	Device	Autodecoded (function of AUTODEC_DIS register bit)	None	Device status is returned (SYSCON1.SELF_PWR and DEVSTAT.R_WK_OK bits).
	Interface	Non-autodecoded	The CPU must stall the command (via SYSCON2.STALL_CMD bit) if interface number is not correct. No feature is defined for interface.	Command is passed to the CPU.
	Endpoint	Autodecoded(function of AUTODEC_DIS register bit)	None	The core automatically stalls the command if endpoint number is different from 0.
CLEAR/SET FEATURE	Device	Autodecoded (function of AUTODEC_DIS register bit)	None (DS_CHG IT is asserted to the CPU after any DEVSTAT.R_WK_OK bit modification).	The core handles the request.
	Interface	Autodecoded (function of AUTODEC_DIS register bit)	None. (No feature is defined in USB 1.1 spec for interface. These requests are stalled).	Command is stalled anyway.
	Endpoint	Non-autodecoded	The CPU must stall the command (via SYSCON2.STALL_CMD bit) if endpoint number/type/direction is not correct. The CPU must reset the EP after having handled the pending transactions (if CLEAR) or set halt condition (if SET). For EP 0, CPU must only clear or set halt condition; FIFO and data PID is always correct for next setup.	Command is passed to the CPU.

<sup>(1)</sup> Transactions on endpoints other than zero are ignored if the device is not configured (addressed state).

<sup>(2)</sup> If some endpoints are not used by the interface currently set, transactions on these endpoints are not ignored; the CPU must set halt feature for the endpoint. This does not happen if USB host works correctly.)

<sup>(3)</sup> If endpoint 0 is halted, per *USB 1.1 Specification* (see 9.4.5: Get\_Status), all requests are stalled except GET\_STATUS, CLEAR\_FEATURE, and SET\_FEATURE requests.)

<sup>(4)</sup> Requests are handled with respect to USB 1.1 when specified as such, but many device reactions are not specified by USB 1.1.)



**Table 28-55. Autodecoded Versus Non-Autodecoded Control Requests <sup>(1)</sup> <sup>(2)</sup> <sup>(3)</sup> <sup>(4)</sup> (continued)**

Request	Recipient	Status	CPU Required Action	Device Behavior if Device Is Not Configured
SET_ADDRESS	Device	Autodecoded	None (see <sup>(5)</sup> ) (Whether the device is addressed or not is available in DEVSTAT register. A valid SET_ADDRESS request with address number from 0 generates a DS_CHG interrupt to the CPU).	Default: device moves in the addressed state if address number is different from 0. Addressed: device takes the new address value or moves in default state if address number is 0. Configured: request is STALLED.
GET_DESCRIPTOR	All	Non-autodecoded	The CPU must write descriptor data into endpoint 0 FIFO.	Command is passed to the CPU.
SET_DESCRIPTOR	All	Non-autodecoded	The CPU must stall the command (via SYSCON2.STALL_CMD bit) if it does not support set descriptor requests.	Command is passed to the CPU.
GET/SET CONFIGURATION	Device	Non-autodecoded	The CPU must stall the command (via SYSCON2.STALL_CMD bit) if configuration number is not correct. If the request is SET_CONFIG, the CPU must reset all endpoints, halt endpoints not used by the default interface setting, set SYSCON1.SELF_PWR value if device is self-powered for the configuration set, and then set SYSCON2.DEV_CFG bit (if config nb is not 0), or set SYSCON2.CLR_CFG bit (if config nb is 0) before allowing status stage to complete. The device moves to configured state (if DEV_CFG set), or moves to addressed state (if CLR_CFG set) and a DS_CHG interrupt is asserted to the CPU.	Command is passed to the CPU.
GET/SET INTERFACE	Interface	Non-autodecoded	The CPU must stall the command (via SYSCON2.STALL_CMD bit) if interface/setting number is not correct. If the request is SET_INTERFACE, the CPU must reset endpoints used by the interface and halt endpoints not used by the interface setting before allowing status stage to complete.	Command is passed to the CPU.
SYNCH_FRAME	Endpoint	Non-autodecoded	The CPU must stall the command if it does not support SYNCH_FRAME request, else write requested data in the endpoint 0 FIFO.	Command is passed to the CPU.

<sup>(5)</sup> During a SET\_ADDRESS autodecoded command, only the seven LSBs are significant for the new address (decimal value from 0 to 127); all others are discarded.

### 28.3.7.6 Note on Control Transfers Data Stage Length

The control transfer data stage length is indicated in the setup data packet.

During control reads, if the USB host requests more data than indicated in the setup packet, unexpected IN transaction is STALLED, causing STALL handshake for all remaining transactions of the transfer until next SETUP. If the USB host requires less data than indicated in the setup packet, the transfer is not STALLED. However, if the host moves to status stage earlier than expected for a non-autodecoded request, the OUT status stage is NAKed because the CPU has not enabled the RX FIFO.

During control writes, if the USB host sends more bytes than indicated in the setup packet, the transfer is STALLED. If the USB host sends fewer bytes than were expected, the request is accepted. But if the USB host moves to status stage earlier than expected for a non-autodecoded request, the IN status stage is NAKed because the CPU has not enabled the TX FIFO.

### 28.3.8 USB Device Initialization

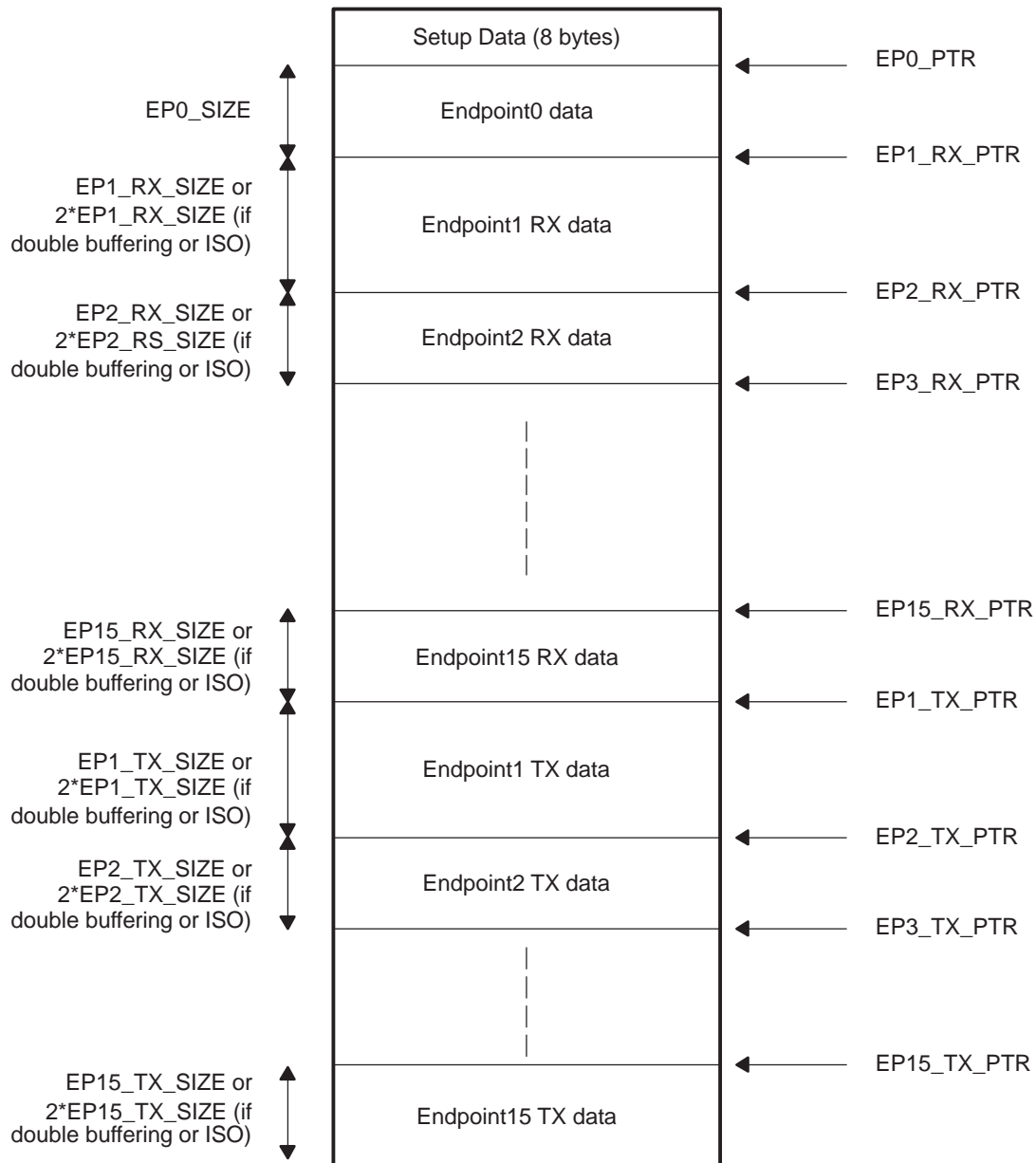
To allow communication between the device and a USB host, the CPU must configure the device by filling the configuration registers.

For each endpoint, the CPU must write on the dedicated register:

- Endpoint size
- Whether double-buffering is allowed for endpoint or not
- Endpoint type (ISO or non-ISO)
- Address of the pointer

System software must choose how to allocate the 2040 available bytes of USB device controller RAM to the USB endpoints. Receive endpoint size and type are configured using the EP1\_RX through EP15\_RX registers. Transmit endpoint size and type are configured using the EP1\_TX through EP15\_TX registers. Figure 28-57 shows an example of the RAM organization, obtained by following the flowchart shown in Figure 28-58.

Figure 28-57. Example of RAM Organization



Once the endpoints are configured, the CPU must set the SYSCON1.CFG\_LOCK bit. If this bit is not set, all transactions are ignored by the core. Then, when the CPU is ready to communicate with the USB host, it must set the SYSCON1.PULLUP\_EN bit. The CPU can wait until the DS\_CHG attach interrupt has been detected and handled before setting the SYSCON1.PULLUP\_EN bit. The USB host cannot detect the device until this bit is set.

Figure 28-58 and Figure 28-59 show flowcharts for the configuration phase.

**Figure 28-58. Device Configuration Routine**

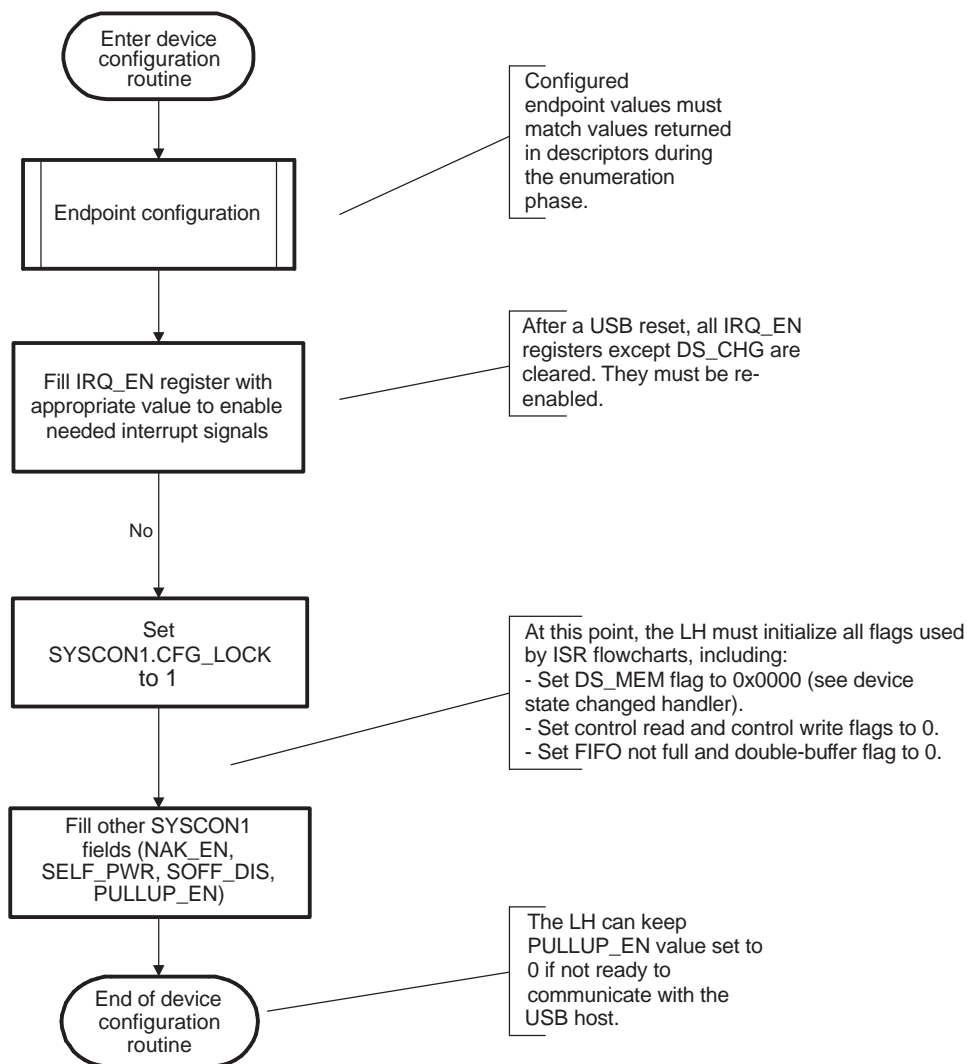
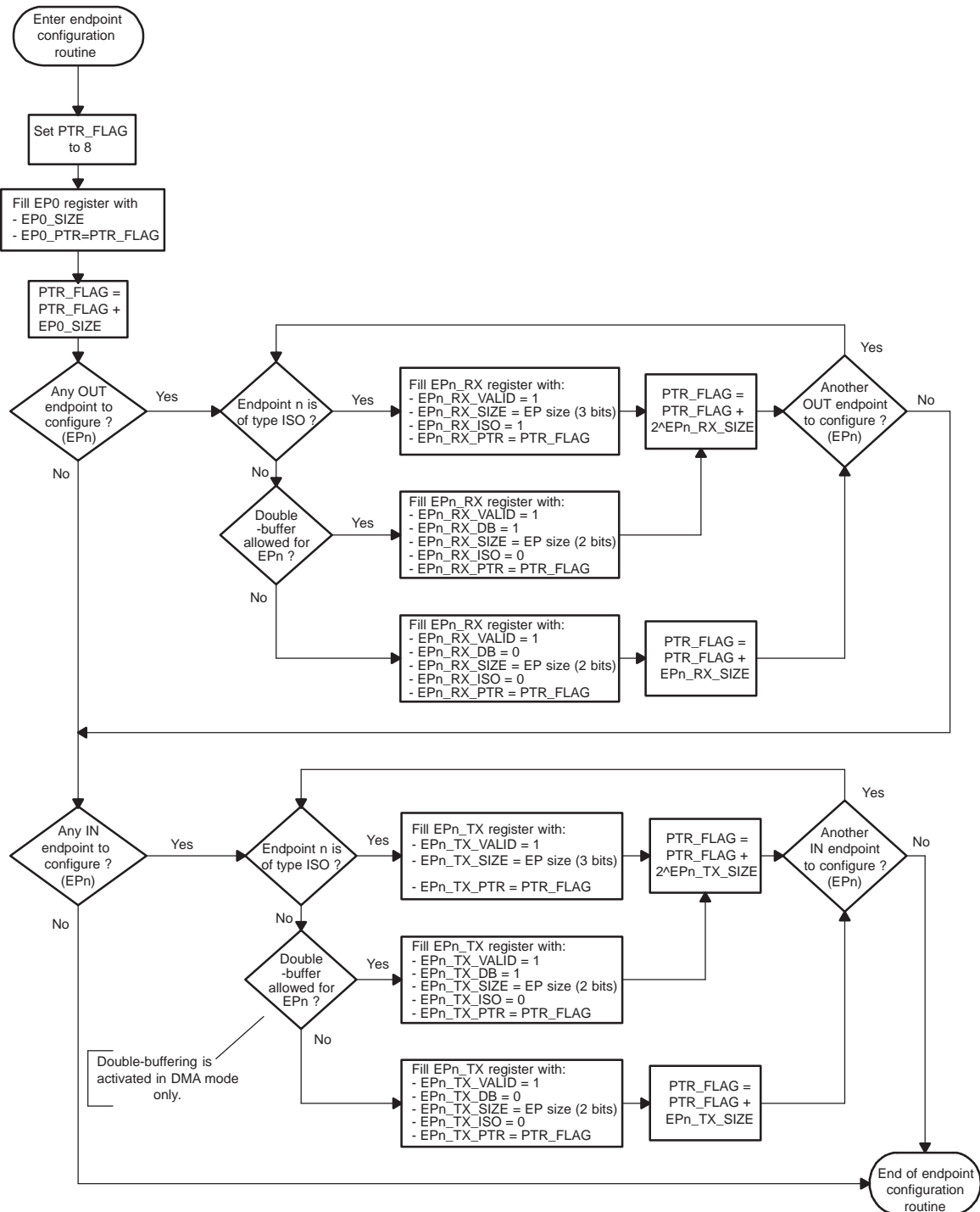


Figure 28-59. Endpoint Configuration Routine



### 28.3.9 Preparing for Transfers

To avoid NAK handshakes for the first transaction on an endpoint, the CPU must prepare the endpoint FIFO for receiving or transferring data. After the first transaction, the FIFO is enabled during the interrupt handling (ISR flowchart).

For receive endpoints, this phase consists of enabling the FIFO to receive data from the USB host. If double buffering is allowed for the endpoint, setting the CTRL.SET\_FIFO\_EN bit enables both FIFOs. Therefore, it is not possible to allow a single transaction when double buffering is used.

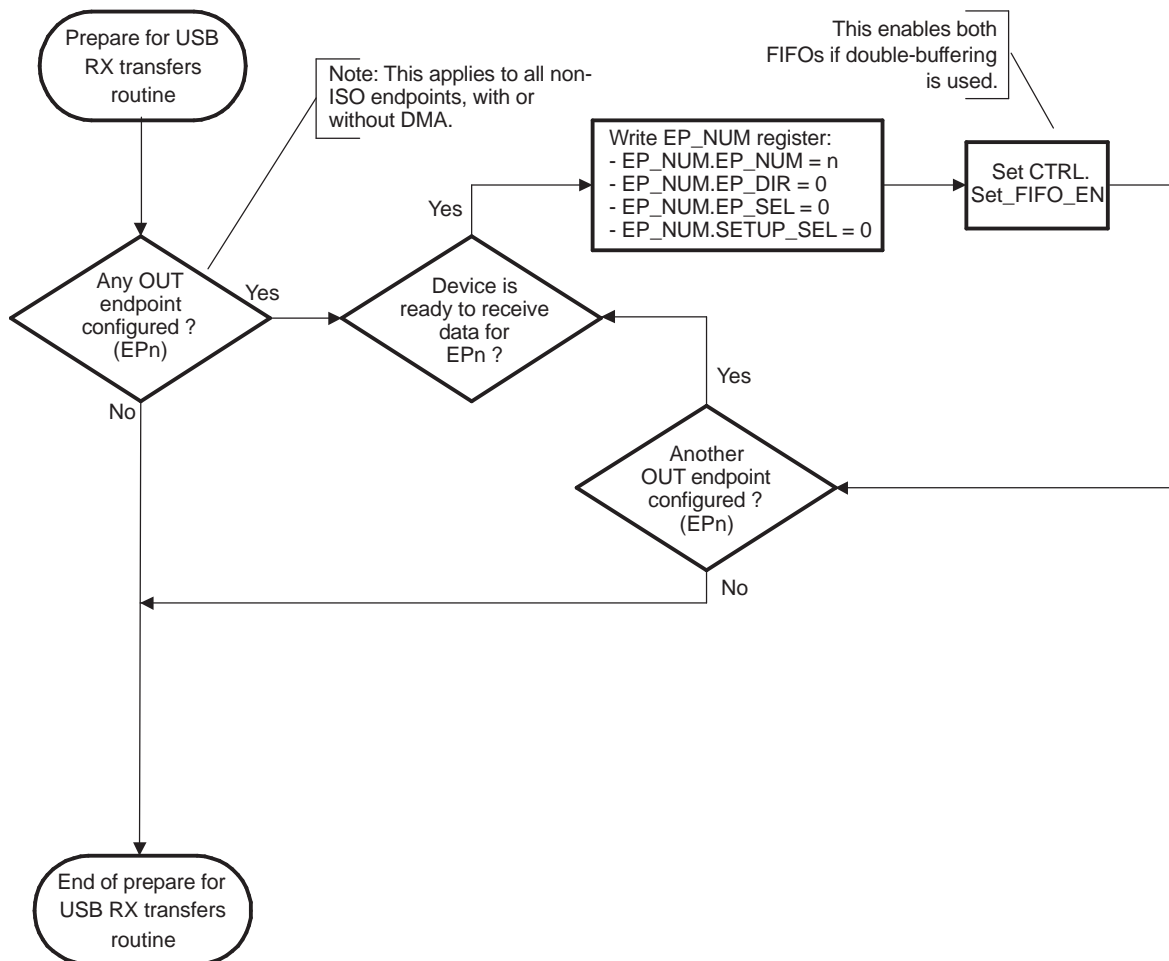
The CPU enters the prepare for USB RX transfers routine, presented once after the enumeration phase, and then properly reacts to EP interrupts. Whether double-buffering is allowed or not is transparent to the CPU, unless both FIFOs are cleared through a CTRL.CLR\_EP or CTRL.RESET\_EP. In that case, and in the case where the CPU finishes to handle an interrupt without having set the CTRL.SET\_FIFO\_EN bit, the CPU must reenter the prepare for USB RX transfers routine.

For transmit endpoints, the CPU enters the prepare for endpoint n TX transfer routine, presented each time a new file must be transmitted from endpoint n to USB host. The CPU must not enter this routine until data written into TX FIFO from the previous transfer have all been received successfully by the USB host (ACK interrupt received), unless TX FIFO is cleared through the CTRL.CLR\_EP or CTRL.RESET\_EP bits (see [Figure 28-60](#) and [Figure 28-61](#)).

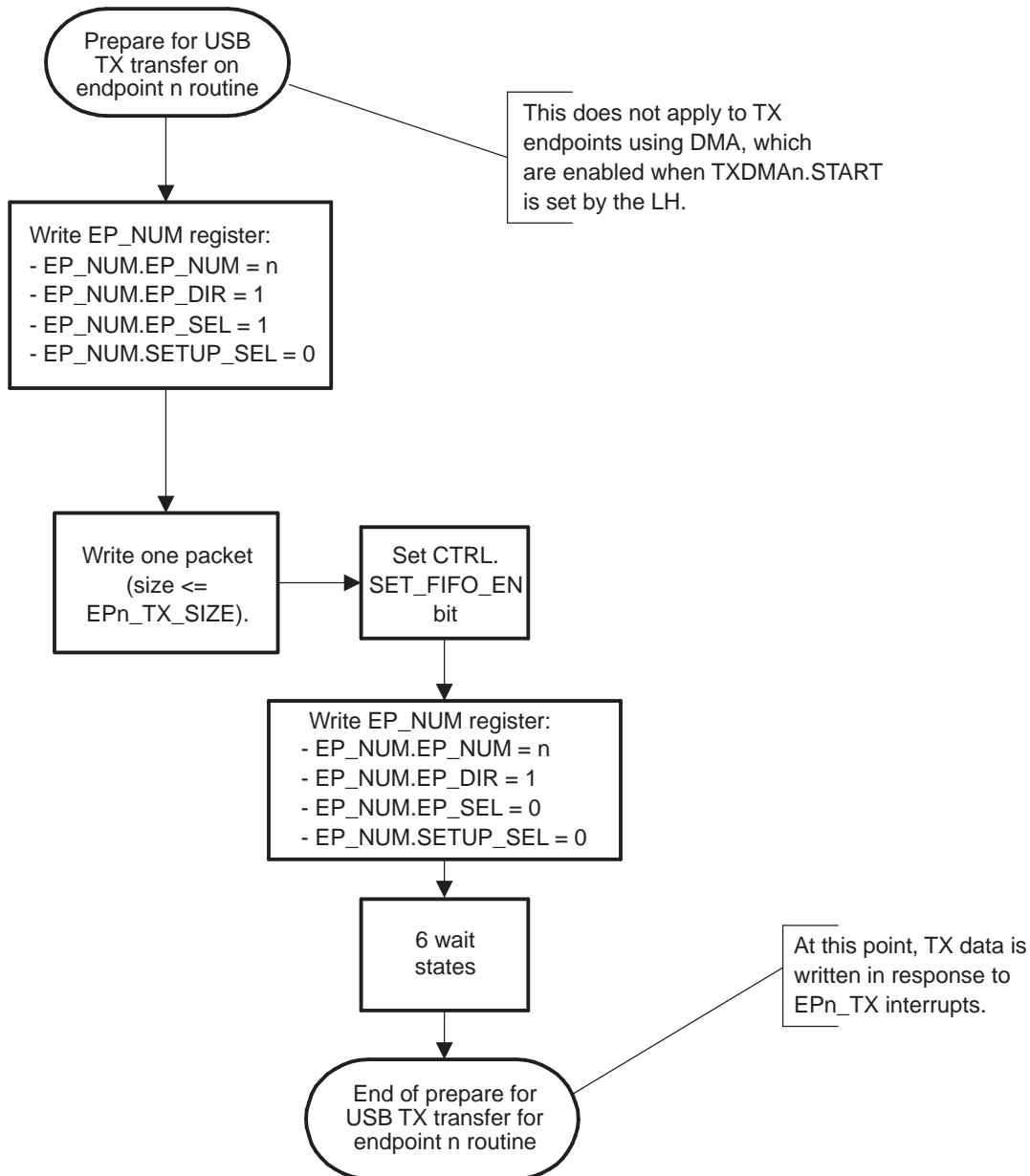
**NOTE:** This does not apply to endpoint 0, which is not used before a setup interrupt occurs. At setup interrupt, the CPU reacts appropriately, and enables EP0 FIFO only if necessary.

To ensure proper usage of the module, you cannot prepare data on different endpoints at the same time. You can enter a routine once the routine is not being used by another endpoint (no parallelism); the EP\_NUM register can be accessed via different routines.

**Figure 28-60. Prepare for USB RX Transfers Routine**



**Figure 28-61. Prepare for TX Transfer on Endpoint n Routine**



### 28.3.10 USB Device Interrupt Service Routine (ISR) Flowcharts

The flowcharts in this section give general operational guidelines for USB device ISR processing. System-architecture-specific details are left to the engineers who write the CPU and USB host code. One USB-specific interrupt register is provided (IRQ\_SRC), including:

- General USB interrupts (including endpoint 0, DMA and device states interrupts) on Vectored Interrupt Manager (VIM) channel # 69 (by default)
- Non-ISO endpoint-specific interrupt on VIM channel # 70 (by default)
- Start of frame (SOF) interrupt for ISO transactions on VIM channel # 68 (by default)

The general USB interrupt ISR must handle non-autodecoded control transfers on endpoint 0 and some specialty interrupts generated because of USB device state modifications or DMA transfers. The ISR for

the endpoint-specific interrupt must handle interrupts from the USB module that are generated because of USB activity for non-isochronous endpoints. The SOF ISR is responsible for handling isochronous endpoints and, if needed by the application, tracking the USB frame number. Many flowcharts are presented in this chapter to provide guidelines for how to handle the interrupts related to the USB device controller module. The flowcharts in this part suppose that the SYSCON1.NAK\_EN bit is cleared.

---

**NOTE:** A key assumption behind the flowcharts presented here is that the application provides separate buffers for each direction of endpoint, except for endpoint 0. The flowcharts read from these application buffers for IN transactions on TX endpoints and write to these application buffers for OUT transactions on RX endpoints.

The USB device controller does not support reentrant interrupts. Each USB device controller must be handled completely before handling another USB device controller interrupt. This restriction occurs because there is only one EP\_NUM register, so endpoint control operations must be completed before working with another endpoint or endpoint direction.

---

### 28.3.11 Important Note on USB Device Interrupts

When an endpoint interrupt is asserted, the CPU writes the EP\_NUM register with the EP\_NUM.EP\_SEL bit set to 1. The CPU must finish the interrupt handling before clearing the EP\_NUM.EP\_SEL bit, because clearing this bit clears the corresponding status bit in the STAT\_FLG register (ACK, NAK, STALL). When an interrupt is pending on an endpoint, the CPU must not select and then unselect the endpoint without handling the interrupt, because this clears the pending transaction status flags. The CPU does not need to set EP\_NUM.EP\_SEL to 1 when setting CTRL.SET\_FIFO\_EN, CTRL.SET\_HALT, and CTRL.CLR\_HALT bits.

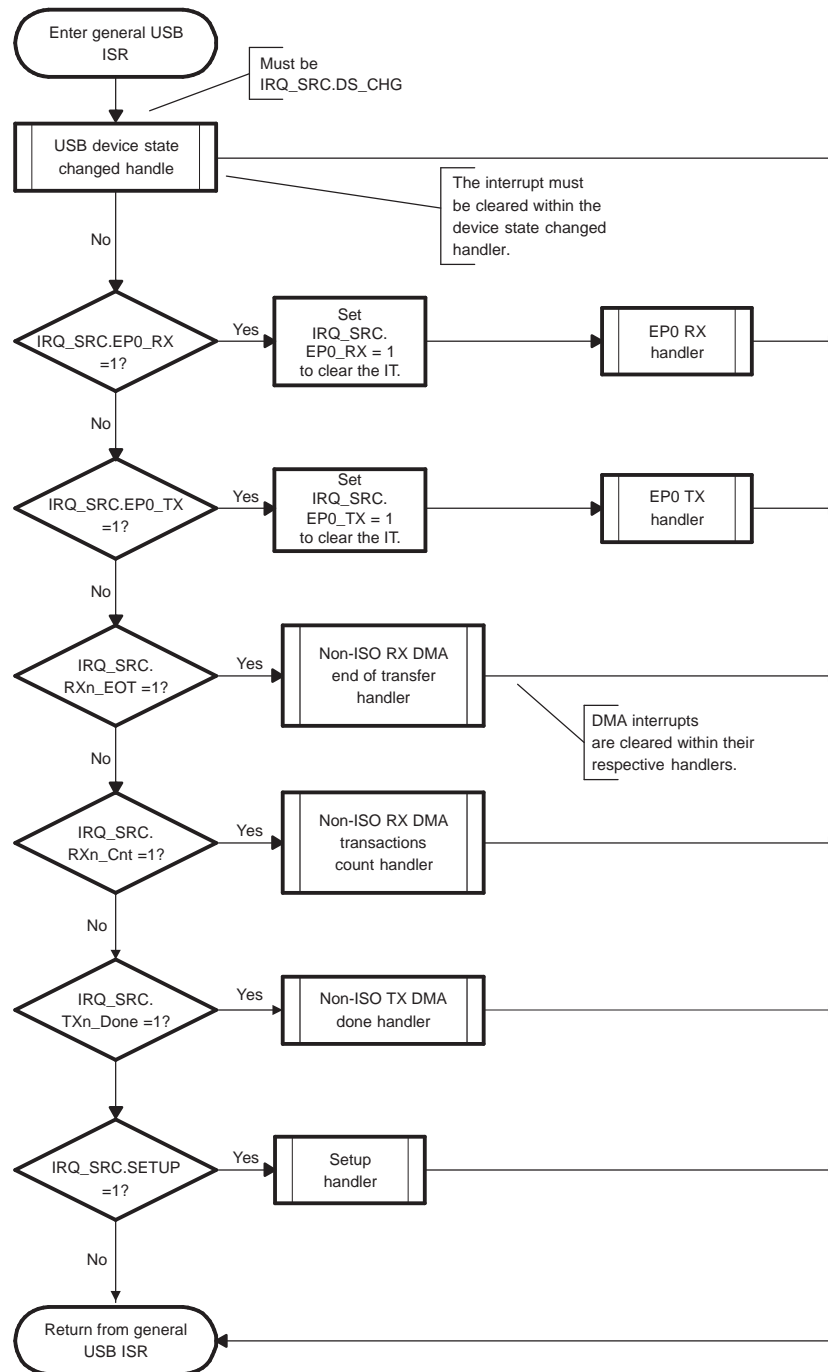
The endpoint status (STAT\_FLG register) is updated at the end of each USB transaction if the previous transaction has been handled. If a pending interrupt has not been handled when a new non-transparent transaction occurs, status flags are not updated (and NAK is returned, even if FIFO was enabled, or STALLED if the EP halt feature was set), so that the CPU never misses an ACKed transaction. If double-buffering is used for an endpoint, STAT\_FLG is updated if there is zero or one interrupt pending for the endpoint, and is not updated if there are already two interrupts pending on the endpoint.

The CPU does not need to set the SYSCON1.NAK\_EN bit during normal operation. However, for debugging process, this bit can be set when the CPU finishes handling an EP interrupt without having set the corresponding CTRL.SET\_FIFO\_EN bit. During TX transaction, if the SYSCON1.NAK\_EN bit is set, the CPU must wait for a NAK interrupt to write the TX data, to avoid a possible conflict caused by the NAK interrupt received while the CPU was writing the TX data.

### 28.3.12 Parsing General USB Device Interrupt

The general USB interrupt (VIM channel # 69 by default) ISR must parse the interrupt identifier register IRQ\_SRC to determine the types of general USB interrupts that are active. These include interrupts relating to USB device state modifications (USB reset, suspend/resume, during enumeration phase) and control transfers on endpoint 0 or non-ISO DMA transfers in either receive or transmit mode. Multiple interrupts can be active at any time, and all must be dealt with by the ISR before returning from the ISR. [Figure 28-62](#) shows an appropriate flowchart for parsing the general USB interrupts.

Figure 28-62. General USB Interrupt ISR Source Parsing Flowchart



### 28.3.13 Setup Interrupt Handler

A separate interrupt flag exists for setup transactions so that the CPU cannot miss a setup transaction, even if it occurs during the data or status phase of another transfer (case of an aborted transfer). The setup parsing function captures the control transfer request information for use in determining which USB bus activity is needed and in controlling how the CPU must generate or respond to the control transfer. This information includes the following:

- bmRequestType
- bmRequest

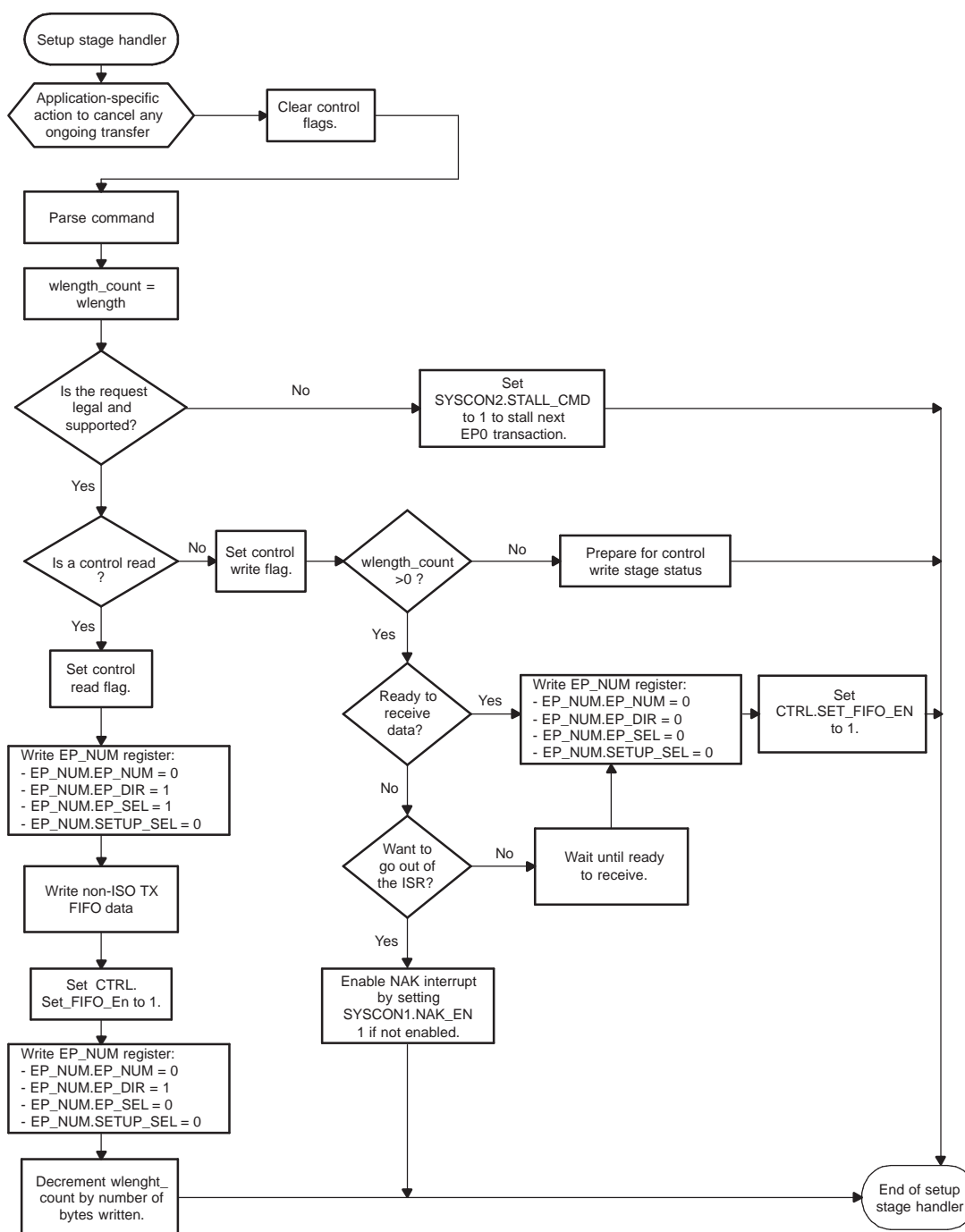


- wValue
- wIndex
- wLength

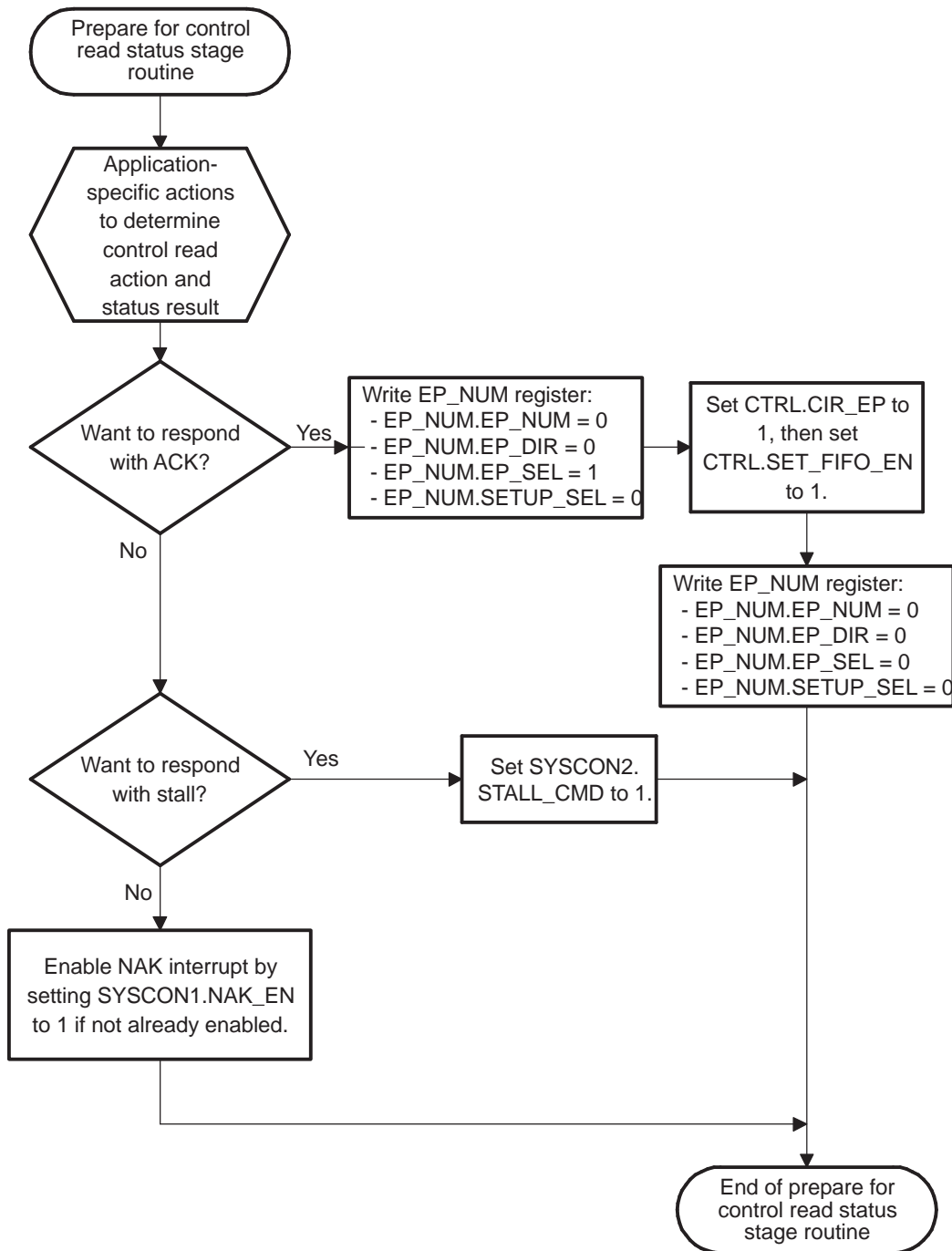
The setup interrupt handler shown in [Figure 28-63](#) is responsible for processing setup transactions occurring on endpoint 0. It calls the routine that parses the control transfer request information, shown in [Figure 28-64](#) to set flags that the rest of the ISR code can use to control proper response to control transfers. Two flags are set by the setup interrupt handler, to be used during endpoint 0 interrupt handlers:

- Control read flag
- Control write flag

**Figure 28-63. Setup Interrupt Handler**



**Figure 28-64. Parse Command Routine (Setup Stage Control Transfer Request)**



**28.3.14 Endpoint 0 RX Interrupt Handler**

Figure 28-65 shows the endpoint 0 RX portion of the general USB interrupt handler, which must handle general USB interrupts related to control OUT transactions on endpoint 0. No EP0 interrupt is generated for autodecoded control transfers.

Figure 28-65. Endpoint 0 RX Interrupt Handler

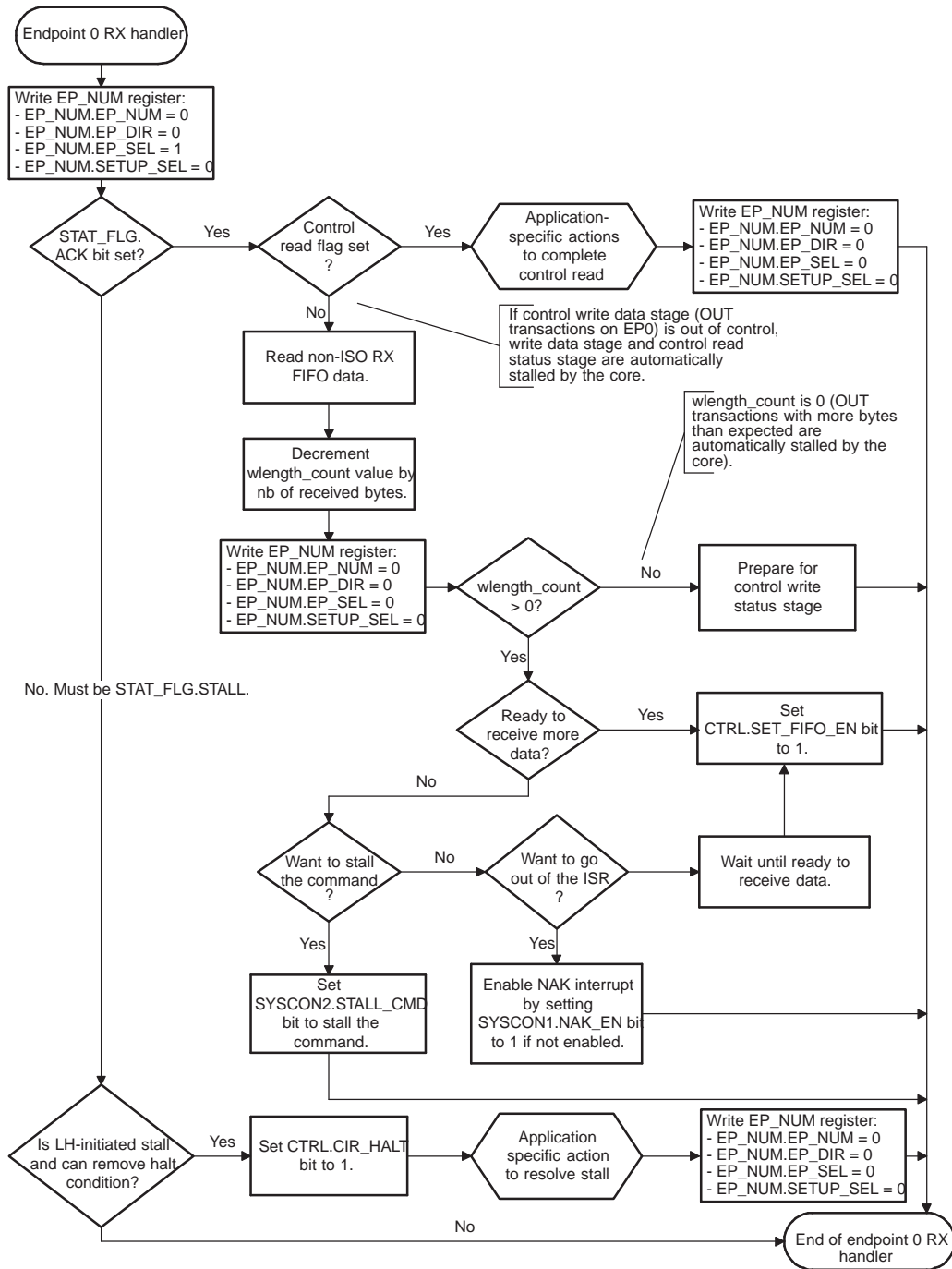
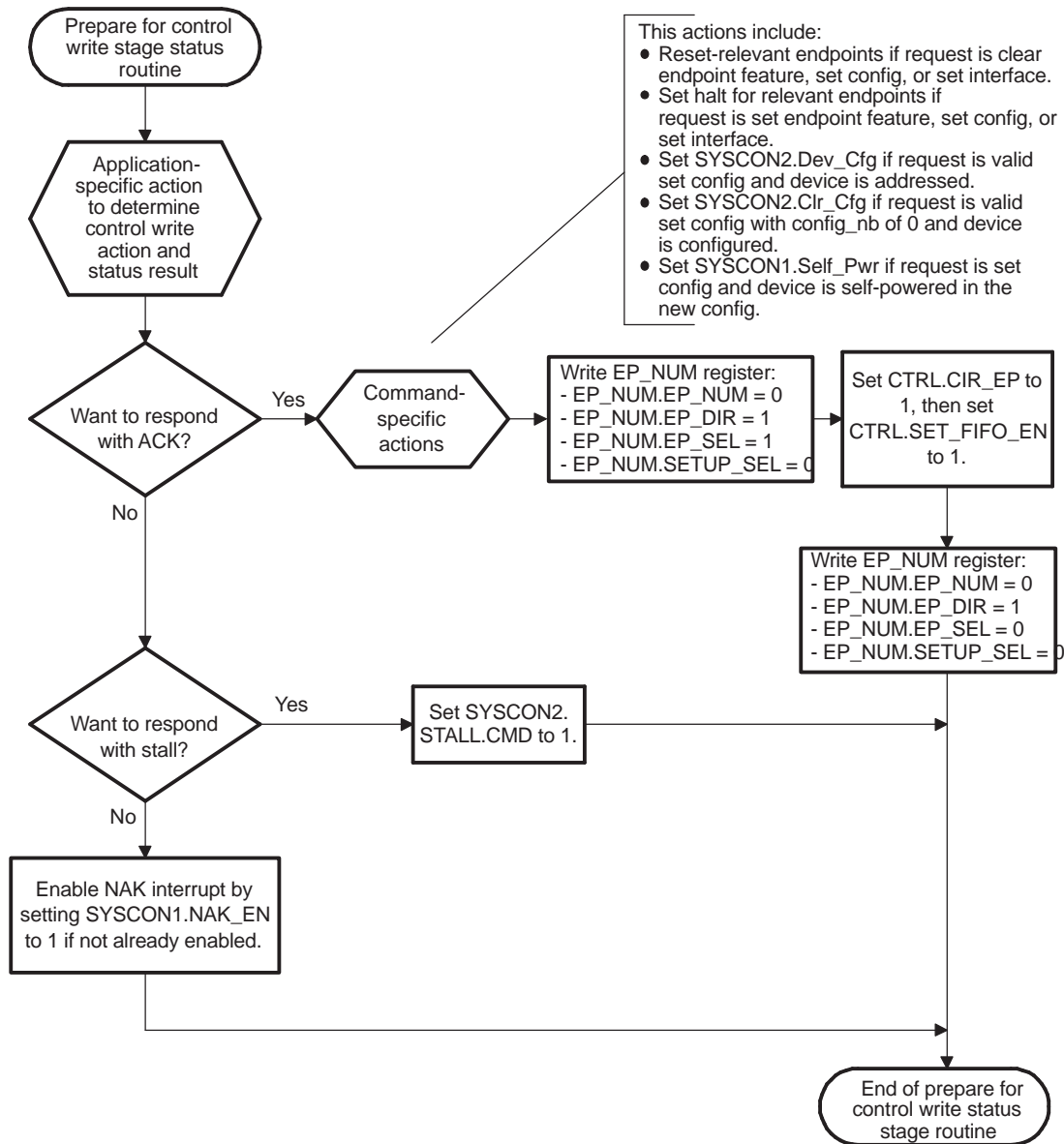


Figure 28-66. Prepare for Control Write Status Stage Routine



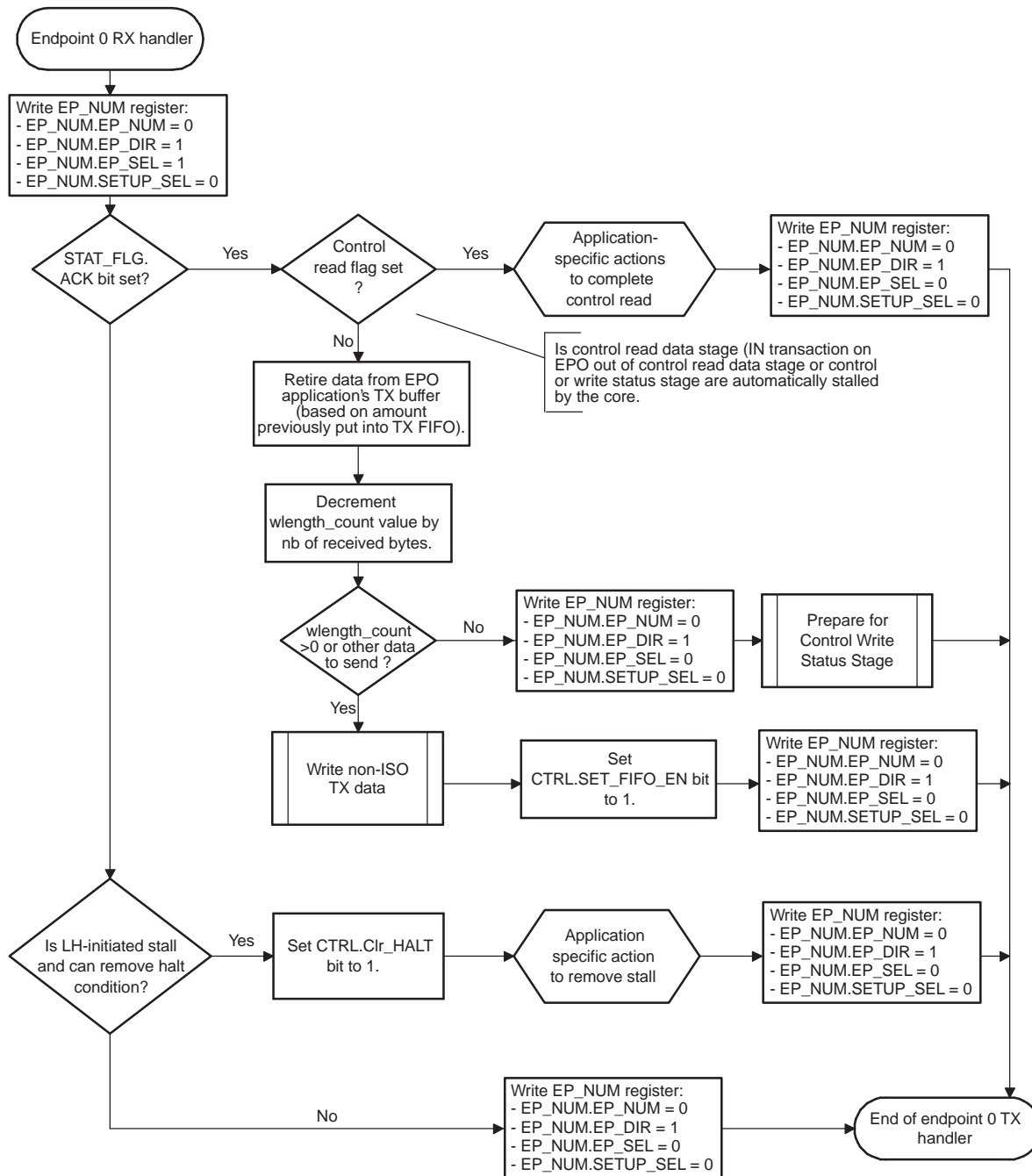
To support the SET\_DESCRIPTOR command, when in commspecific actions, you must first reset all endpoints by selecting the endpoint and then clear it, unlock the configuration, change the new one, lock the configuration, go in prepare for transfers, and enable all interrupts (see Figure 28-66).

### 28.3.15 Endpoint 0 TX Interrupt Handler

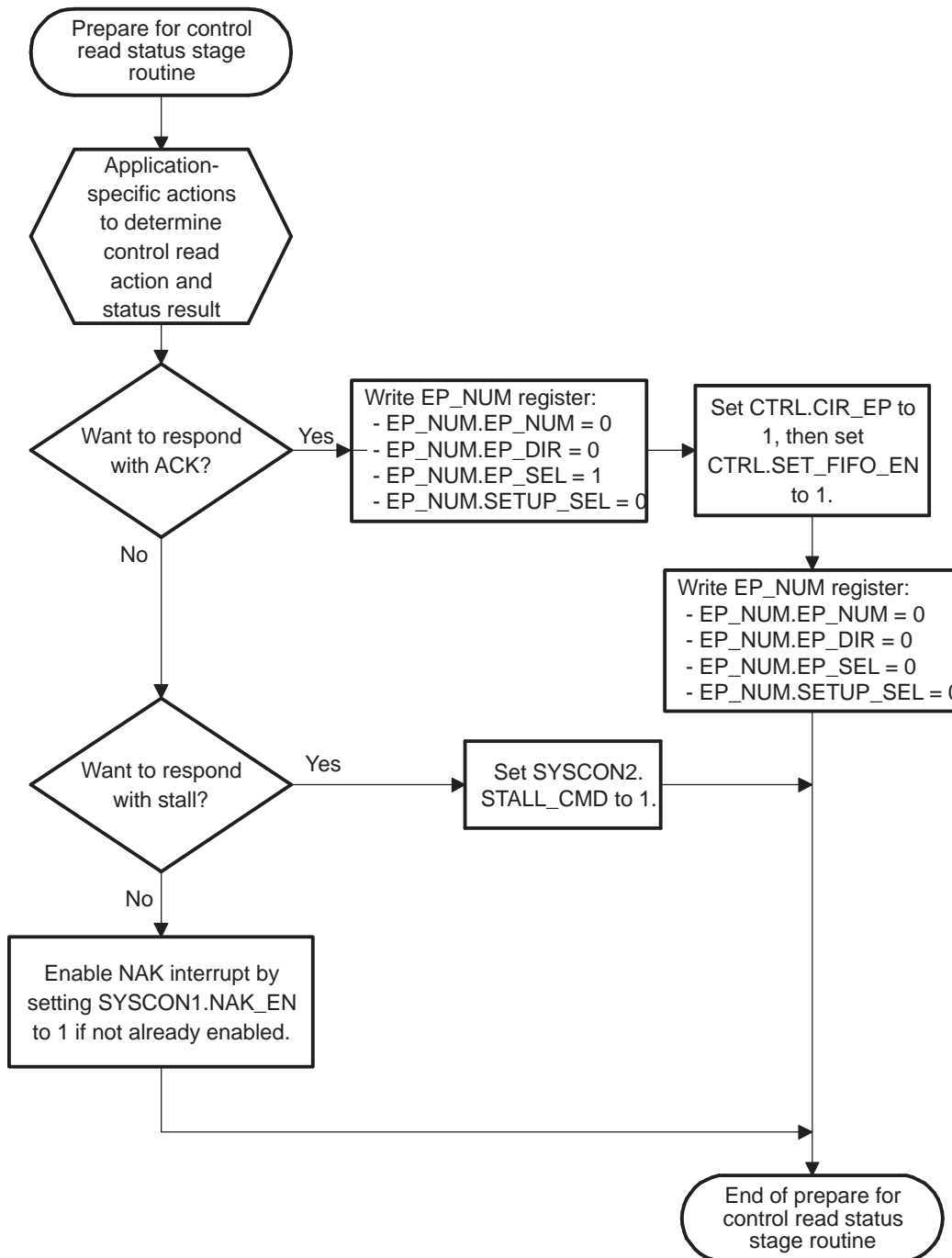
Figure 28-67 shows the TX portion of the general USB interrupt handler, which must handle general USB interrupts related to control IN transactions on endpoint 0.

The endpoint 0 TX interrupt handler must be able to move data into the endpoint 0 TX FIFO when the application buffer for endpoint 0 TX data is not empty and an endpoint 0 TX interrupt occurs signaling an ACKed non-autodecoded endpoint 0 IN transaction. This data can be control read data stage information or control write status stage handshaking information (see Figure 28-68).

Figure 28-67. Endpoint 0 TX Interrupt Handler



**Figure 28-68. Prepare for Control Read Status Stage Routine**



### 28.3.16 Device States Changed Handler

This section describes how USB device states and transitions states are decoded by the USB device controller and how they can be handled.

The state machine (see [Figure 28-69](#)) shows how the USB device controller device moves from one state to another state with respect to the USB1.1 specification. Attach/unattach transition is not shown in the transition flow.

Because SET\_CONFIGURATION is not decoded by the core, the CPU has the responsibility to distinguish a SET\_CONFIGURATION with a nonvalid configuration value from other SET\_CONFIGURATION requests and to set SYSCON2.DEV\_CFG only if the configuration value is valid (value 0 is nonvalid), when the device is in addressed state. When the device is in configured state, the CPU has the responsibility to set SYSCON2.CLR\_CFG if the configuration number is 0 so that the device moves to addressed state.

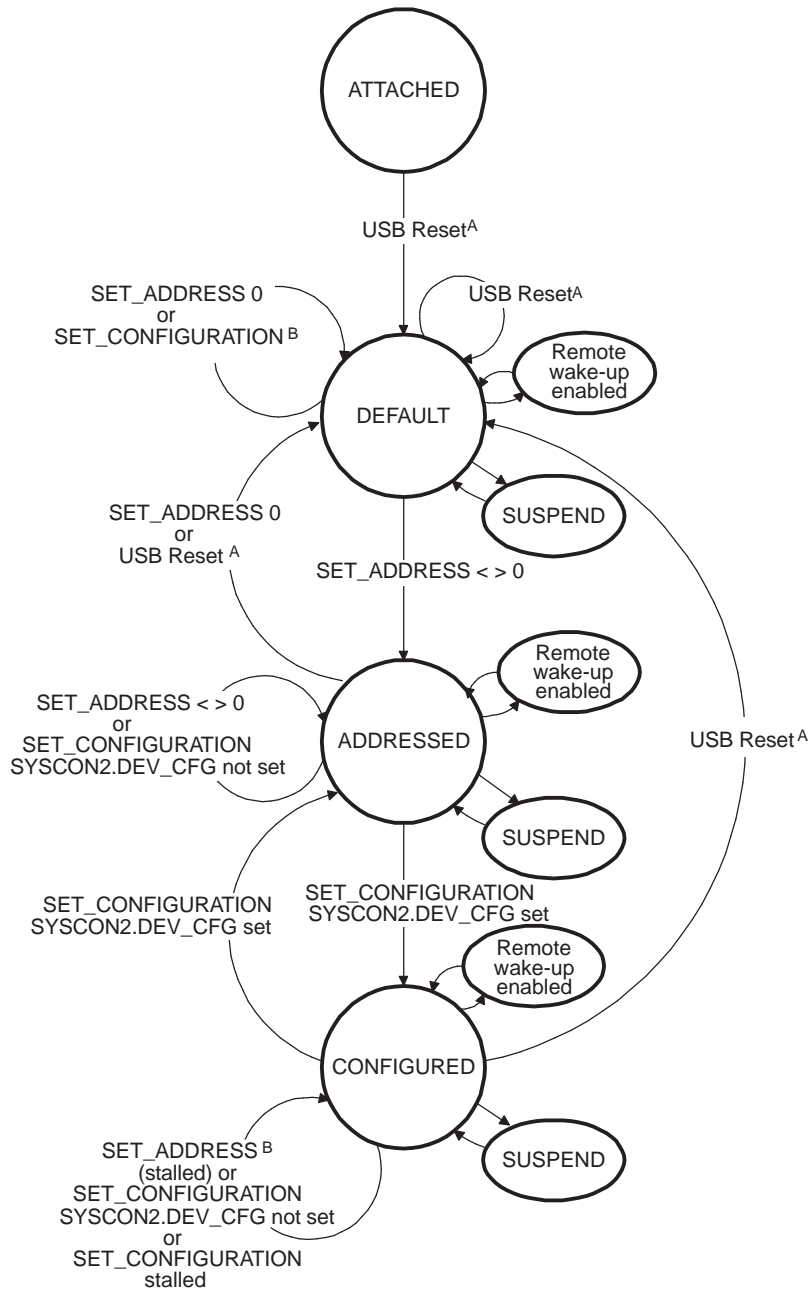
Device states are visible in the DEVSTAT register and are decoded as follows.

- Attached: The device is attached to the USB and powered.
- Default: The device is attached to the USB, is powered, and has been reset.
- Addressed: The device is attached to the USB, is powered, has been reset, and an address has been assigned. The device moves into the addressed state after a SET\_ADDRESS request with an address number other than 0.
- Configured: The device is attached to the USB, is powered, has been reset, has an address other than 0, and is configured. The device moves into the configured state after a valid SET\_CONFIGURATION request, only if the CPU has set the SYSCON2.DEV\_CFG bit (meaning the configuration is valid).
- Suspended: The device is at minimum default and has not seen bus activity for 5 ms.
- Reset: When set, the device is receiving a valid USB host reset.
- R\_WK\_OK: This bit is set (cleared) automatically after a valid SET\_DEVICE\_FEATURE (CLEAR\_DEVICE\_FEATURE) request from the USB host.

Any change in the DEVSTAT register bits triggers a device change interrupt (IRQ\_SRC.DS\_CHG), if enabled.

The device moves to addressed state after the status stage of a valid SET\_ADDRESS, even if the status stage ACK handshake is received corrupted or not sent by the USB host. A SET\_DEVICE\_FEATURE or a CLEAR\_DEVICE\_FEATURE is effective after setup transaction, even if no status stage occurs. A SET\_CONFIGURATION request is effective before status stage, when the CPU sets the SYSCON2.CLR\_CFG or SYSCON2.DEV\_CFG bit (see [Figure 28-70](#)).

Figure 28-69. USB Device Controller Device State Transitions



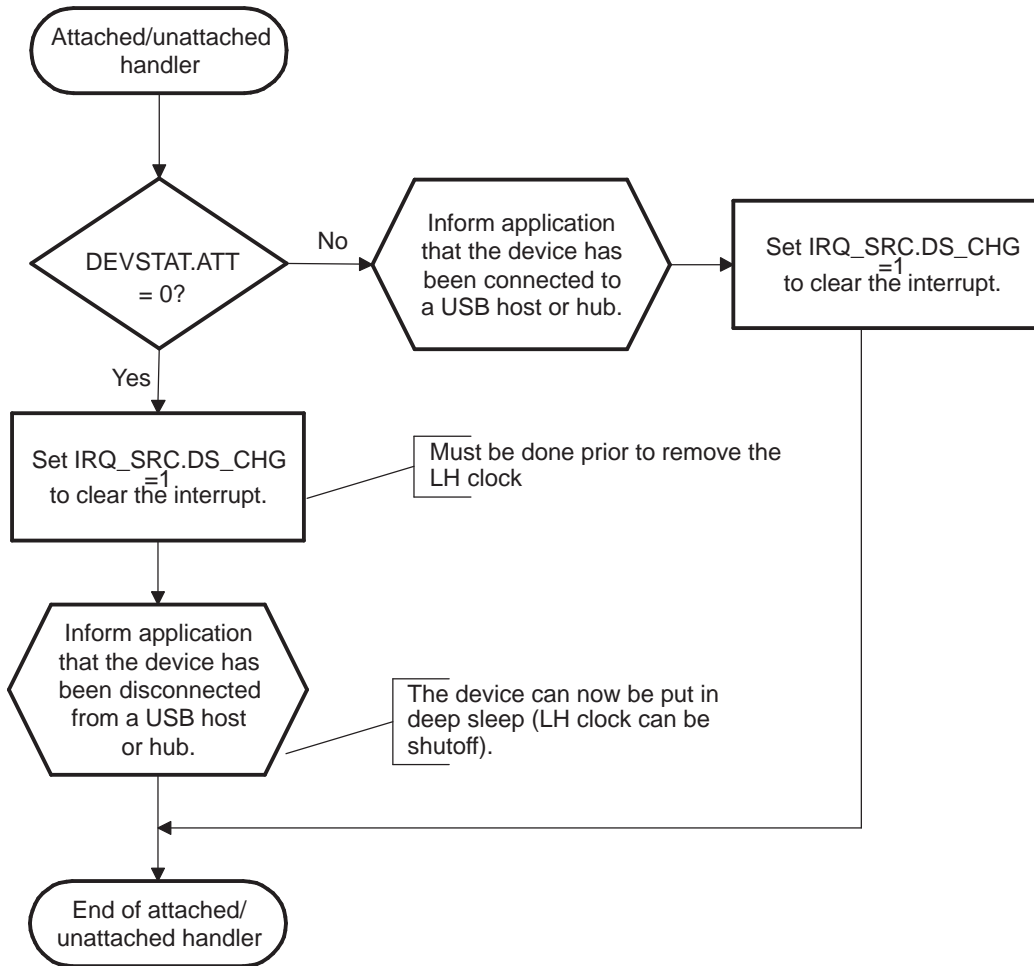
Behavior not specified by USB 1.1 specifications

- A USB reset generates two interrupts (when USB reset is asserted and then when USB reset completes.)
- B No interrupt is asserted by the core for transitions shown with dashed lines.



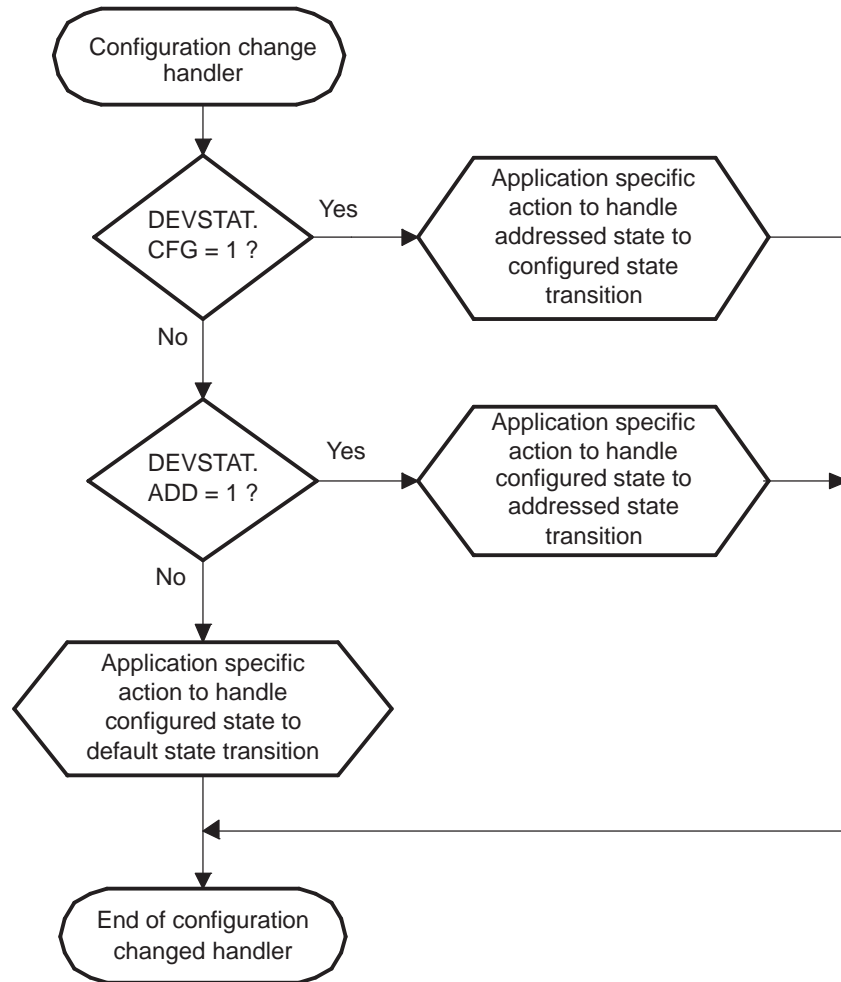


Figure 28-71. Attached/Unattached Handler



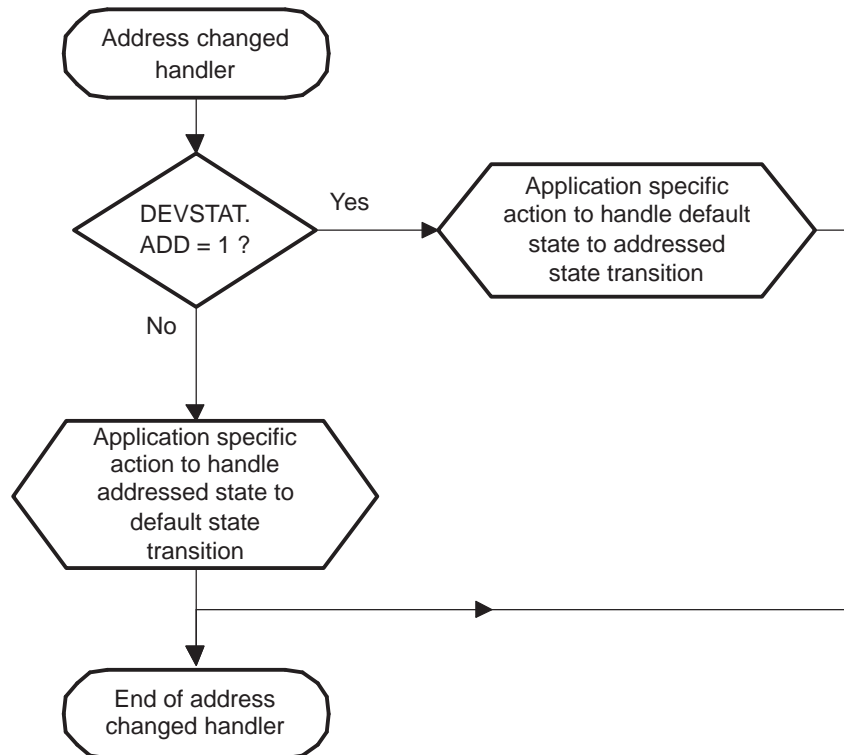
**28.3.18 Device State Configuration Changed Handler**

When a configuration changed interrupt occurs, the USB device has received a set configuration operation. When this occurs, the configuration-changed handler performs the operations shown in [Figure 28-72](#).

**Figure 28-72. Configuration Changed Handler****28.3.19 Device State Address Changed Handler**

When a address changed interrupt occurs, the USB device has received a set address operation. When this occurs, the address-changed handler performs the operations shown in [Figure 28-73](#).

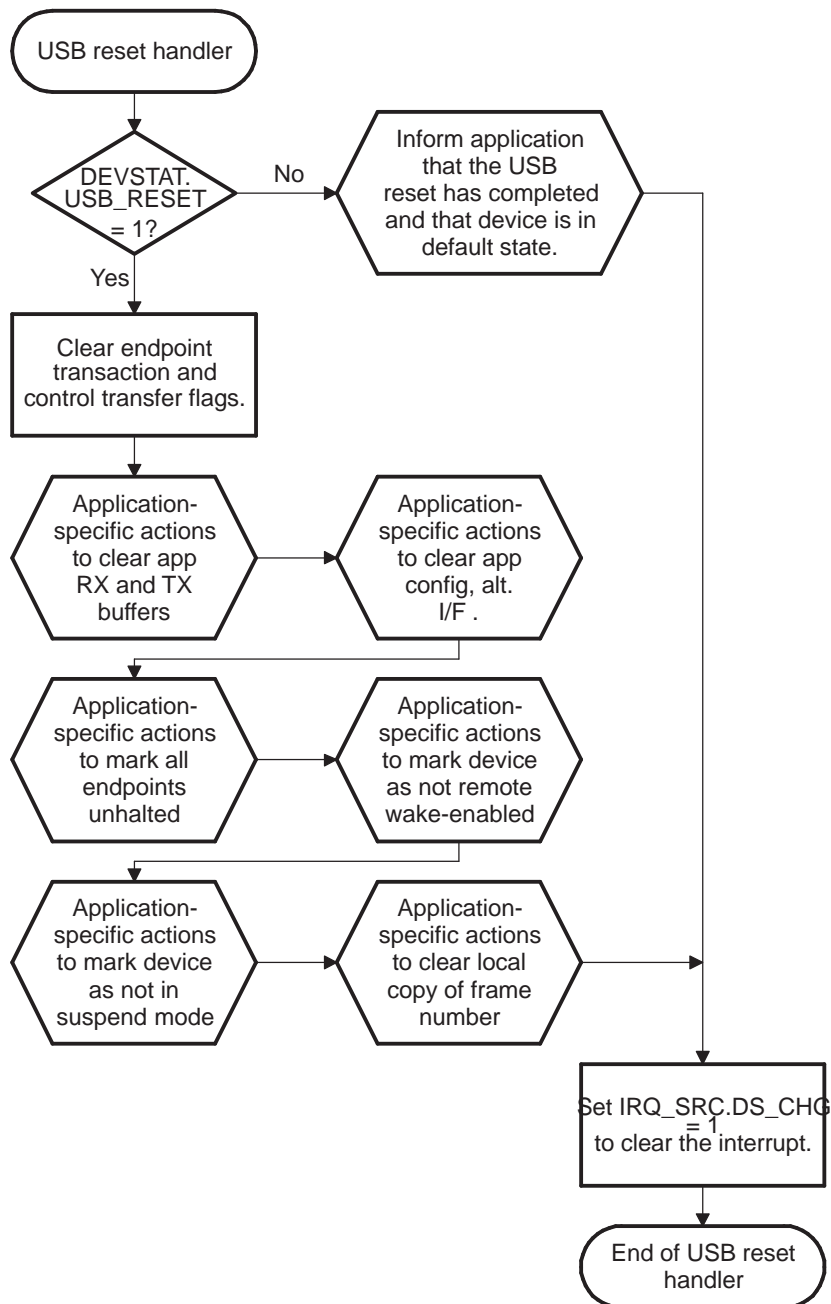
**Figure 28-73. Address Changed Handler**



### 28.3.20 USB Device Reset Interrupt Handler

When a USB reset occurs, the USB module generates a general USB interrupt to the CPU (see [Figure 28-74](#)). The CPU responds to this interrupt by performing the following operations:

- Cancels any ongoing USB transaction and/or control transfer handling
- Clears any copies that the application has of configuration number or alternate interface numbers
- Clears any application-specific information relating to halted endpoints
- Clears any application-specific information relating to the remote wake enable flag
- Clears any application-specific information relating to the suspend mode flag
- Clears any application-specific copy of the frame number

**Figure 28-74. USB Device Reset Handler Flowchart**

### 28.3.21 Suspend/Resume Interrupt Handler

When a USB device suspend/resume general USB interrupt occurs, the USB module has either entered or left suspend mode. The CPU code must determine which and react appropriately (see [Figure 28-75](#)).

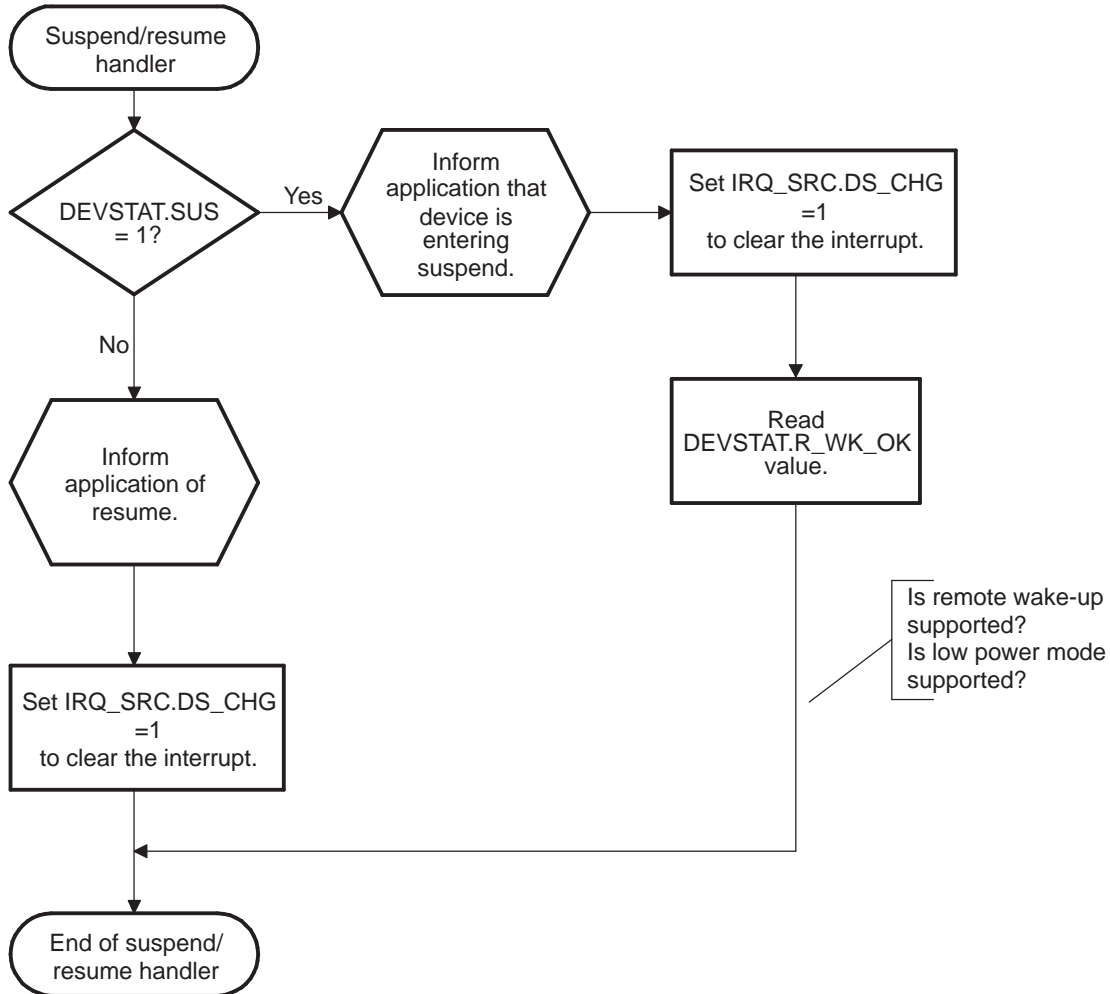
The suspend sense hardware is implemented to trigger only after 5 ms of bus idle. This forces compliance with the *USB Specification Version 1.1*  $T_{WTRSM}$  timing parameter (3 ms of IDLE to identify suspend, 2 ms before the remote device can signal resume).

If the CPU wants to wake the device from suspend mode and remote wake-up enable is set ( $DEVSTAT.R\_WK\_OK = 1$ ), it must first turn its clock on (if stopped), and then set  $SYSCON2.RMT\_WKP$ . The device then drives resume.

If shutoff is enabled (SYSCON1.SOFF\_DIS=0), the 48-MHz clock is automatically shut off at suspend and turned on at resume (USB host or CPU driven). Setting or not setting the SYSCON1.SOFF\_DIS bit is part of the device configuration. However, the CPU can modify its value at suspend interrupt time if necessary.

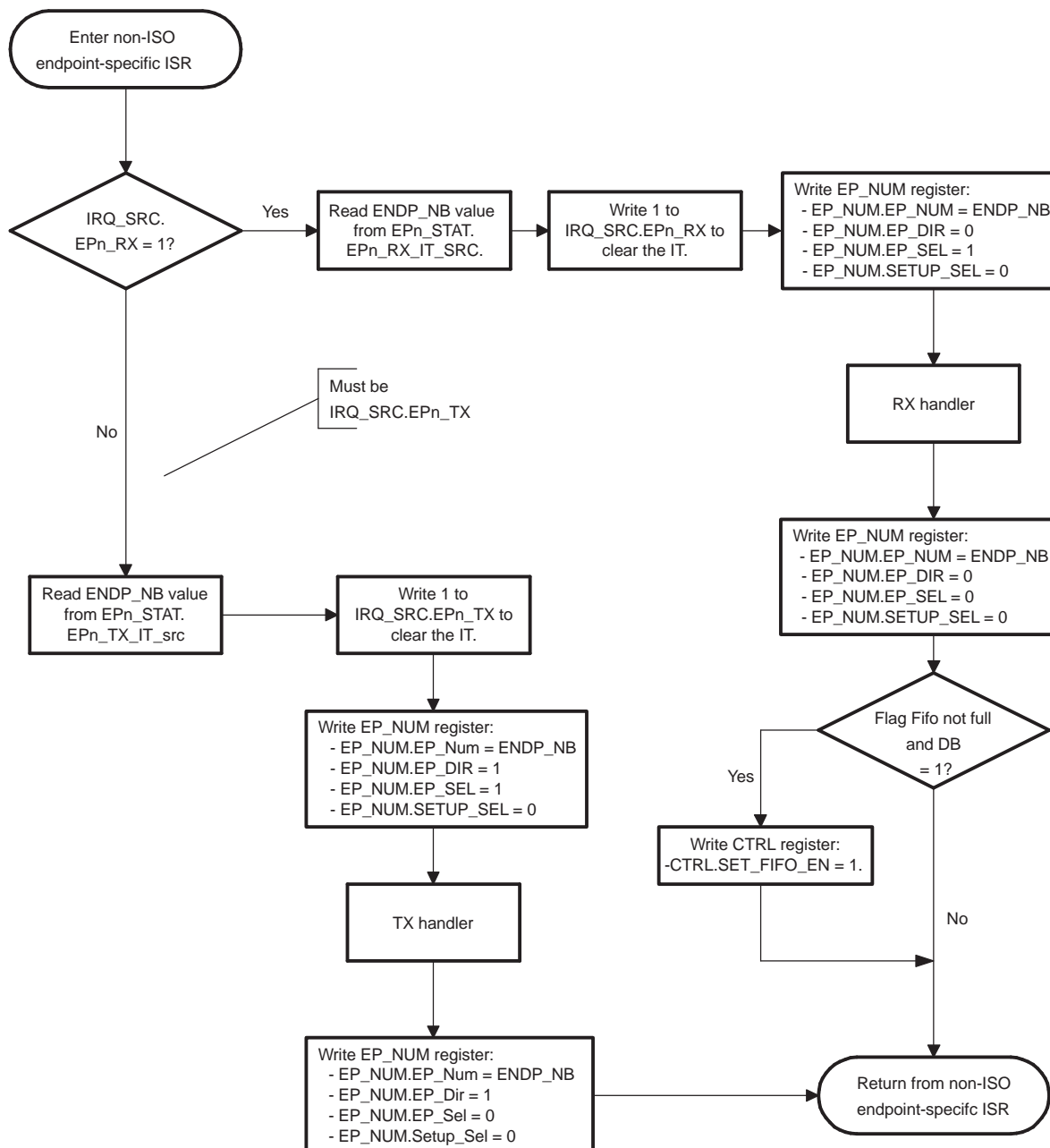
A USB reset is also a valid way to exit suspend mode. But the suspend/resume handler and the USB reset handler do not have to take this into account, because three interrupts are generated in that case (one for resume, one for reset, and one for end of reset).

**Figure 28-75. Typical Operation for USB Suspend/Resume General USB Interrupt Handler**



**28.3.22 Parsing Non-ISO Endpoint-Specific Interrupt**

The endpoint-specific interrupt ISR (also known as the non-isochronous interrupt, on VIM channel # 70 by default) must parse the interrupt identifier register IRQ\_SRC to determine the interrupts that are active (IRQ\_SRC.EPN\_RX, IRQ\_SRC.EPN\_TX, or both). Two interrupts can be active at any time. The ISR must then read the EPN\_STAT register to determine the endpoint causing the interrupt. For each direction, only one endpoint interrupt can be active at a time (see Figure 28-76).

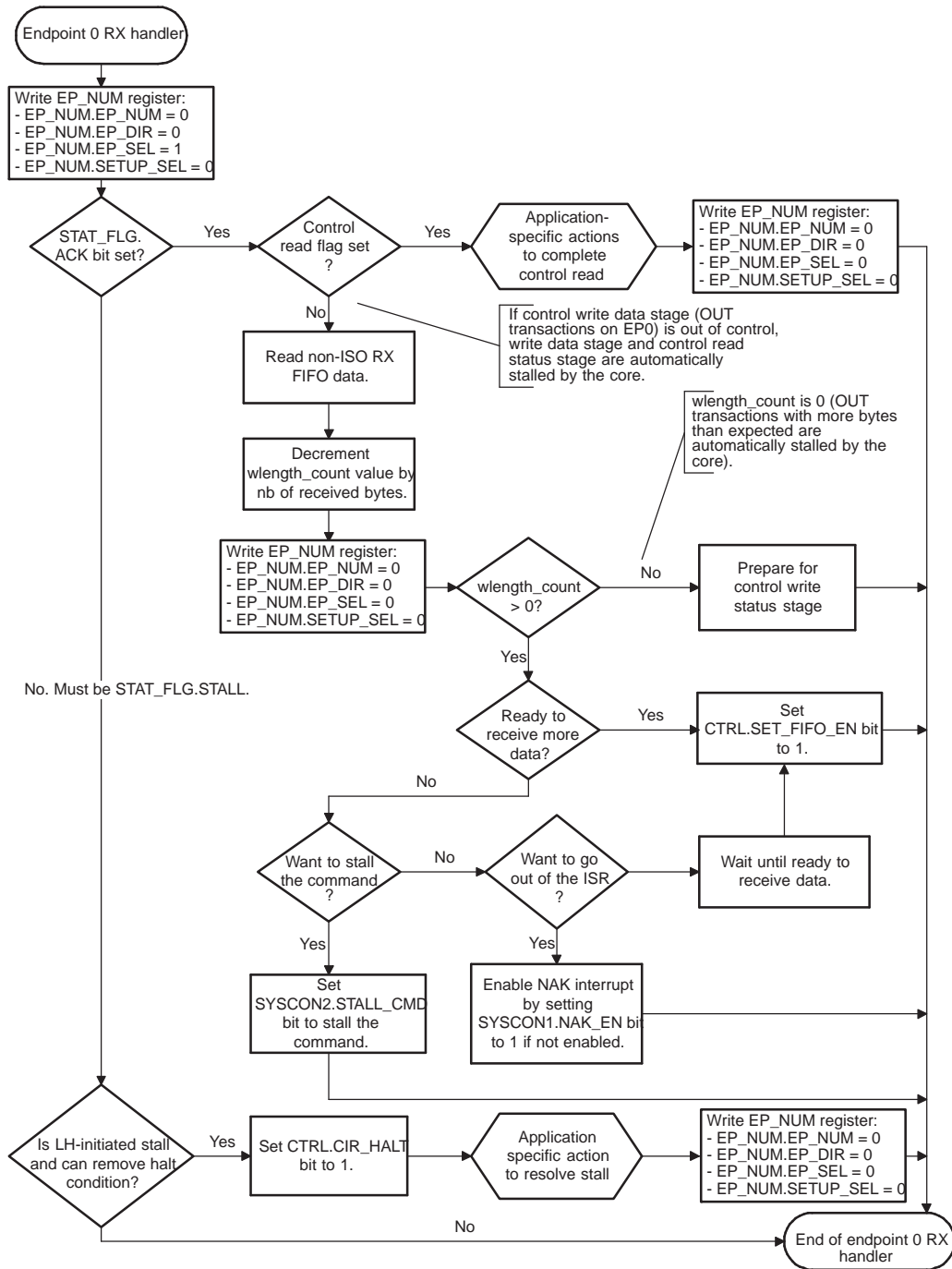
**Figure 28-76. Non-ISO Endpoint-Specific (Except EP 0) ISR Flowchart**

### 28.3.23 Non-ISO, Non-Control OUT Endpoint Receive Interrupt Handler

Figure 28-77 shows the operations necessary to handle non-ISO, non-control OUT endpoint-specific receive interrupts. This flowchart shows two different RX transaction handshaking interrupts. There is a third interrupt handshaking possibility when NAK interrupts are enabled, which is not depicted here. Depending on the application-specific actions needed for various endpoints in the real system, it is possible to use one routine that is common to all of the non-ISO, non-control receive endpoints, where the only differences are in the EP\_NUM register value set and the selection of the proper application RX data buffer in the read non-ISO RX FIFO data routine (see Figure 28-78).

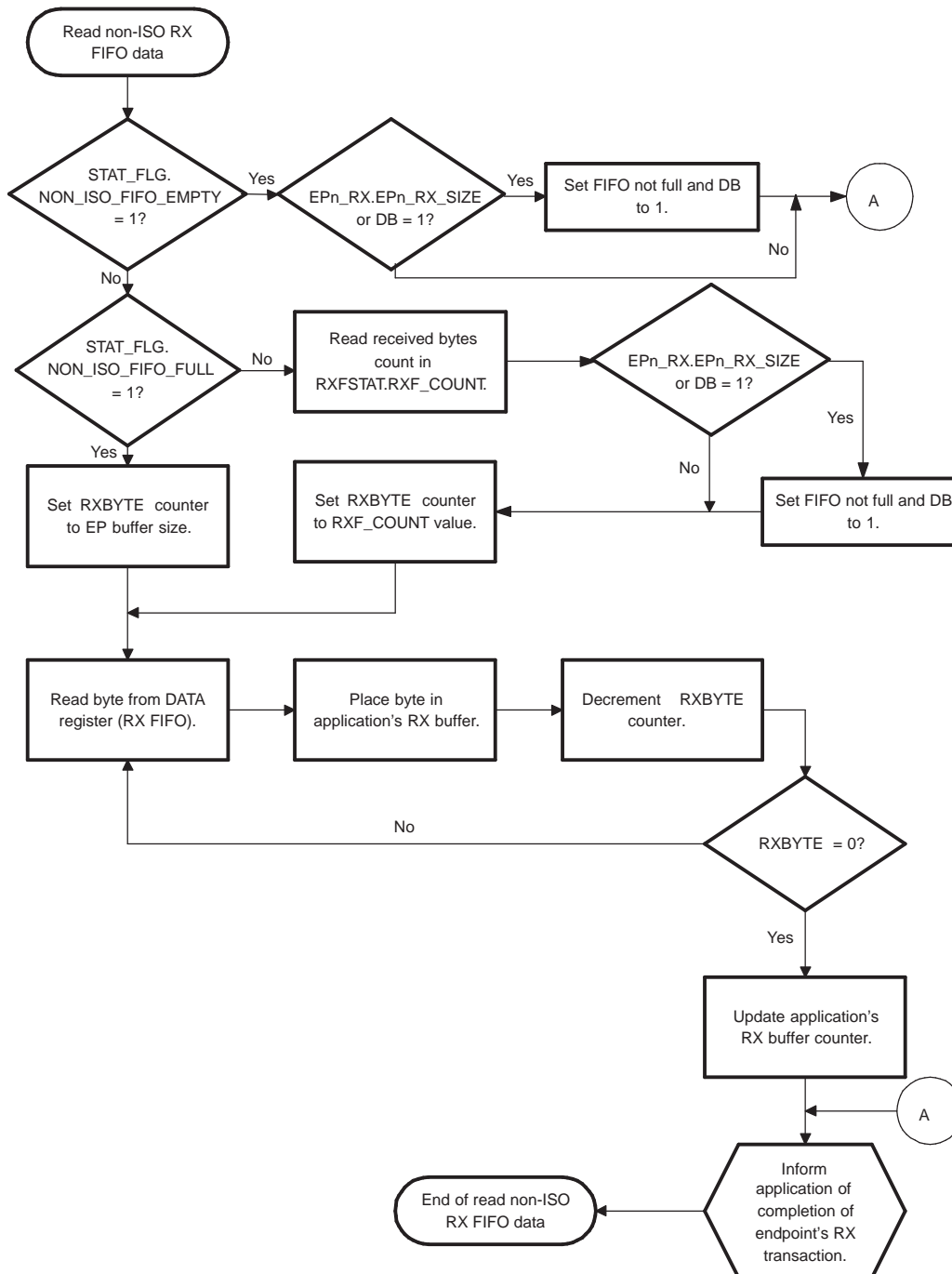
This flowchart does not attempt to document control endpoint 0 receive interrupts, which are discussed separately because of the more complex three-stage transfer mechanism used for control writes.

Figure 28-77. Non-Isochronous Non-Control Endpoint Receive Interrupt Handler





**Figure 28-78. Read Non-Isochronous RX FIFO Data Flowchart**

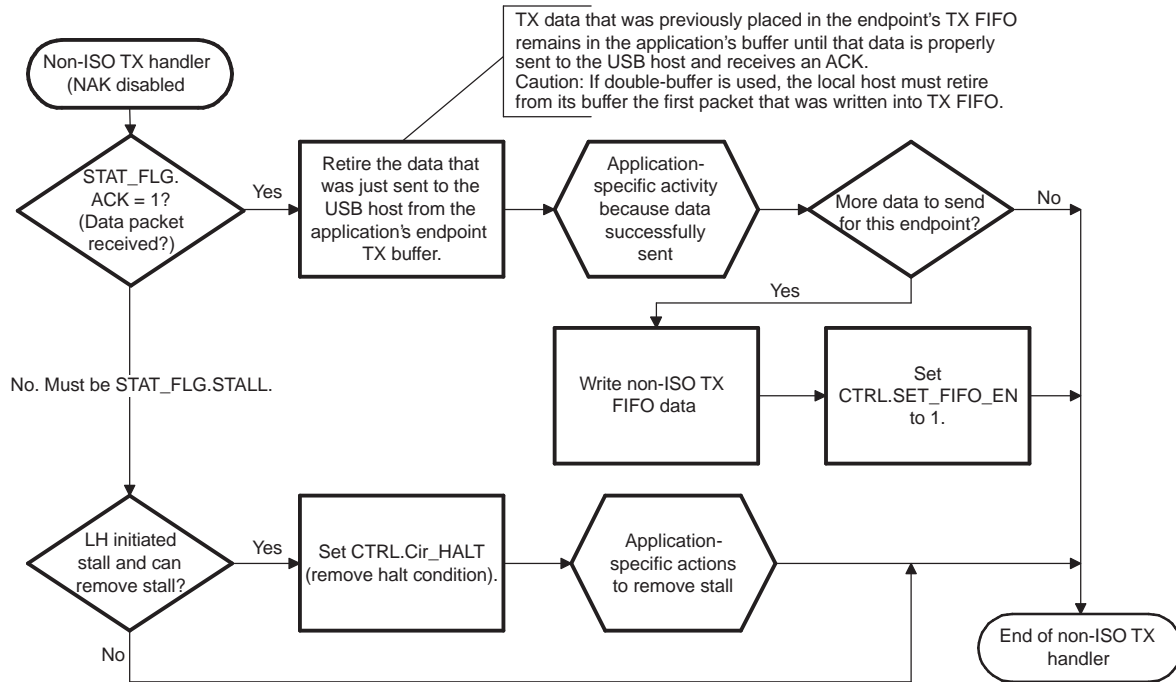


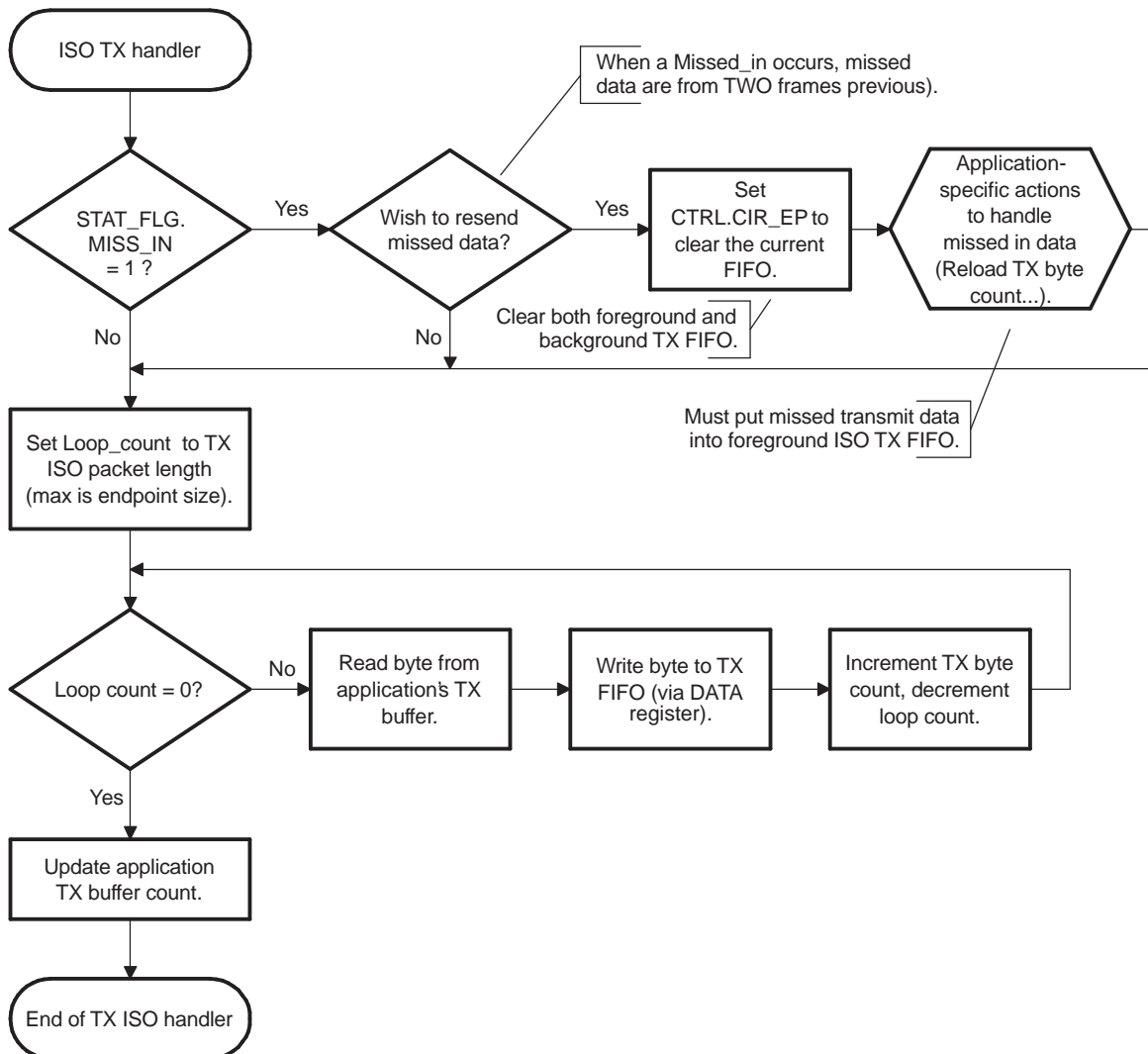
**28.3.24 Non-ISO, Non-Control IN Endpoint Transmit Interrupt Handler**

Figure 28-79 shows the operations necessary to handle non-ISO, non-control IN endpoint-specific transmit interrupts. This flowchart shows two TX transaction handshaking interrupts. There is a third interrupt handshaking possibility when NAK interrupts are enabled, which is not depicted here. Depending on the application-specific actions needed for various endpoints in the real system, it is possible to use one routine that is common to all of the non-ISO, non-control transmit endpoints, where the only differences are in the EP\_NUM register value set and in the routine selection of the application TX buffer (see Figure 28-80).

This flowchart does not attempt to document control endpoint 0 transmit interrupts, which are discussed separately because of the more complex three-stage transfer mechanism used for control reads.

**Figure 28-79. Non-Isochronous Non-Control Endpoint Transmit Interrupt Handler**

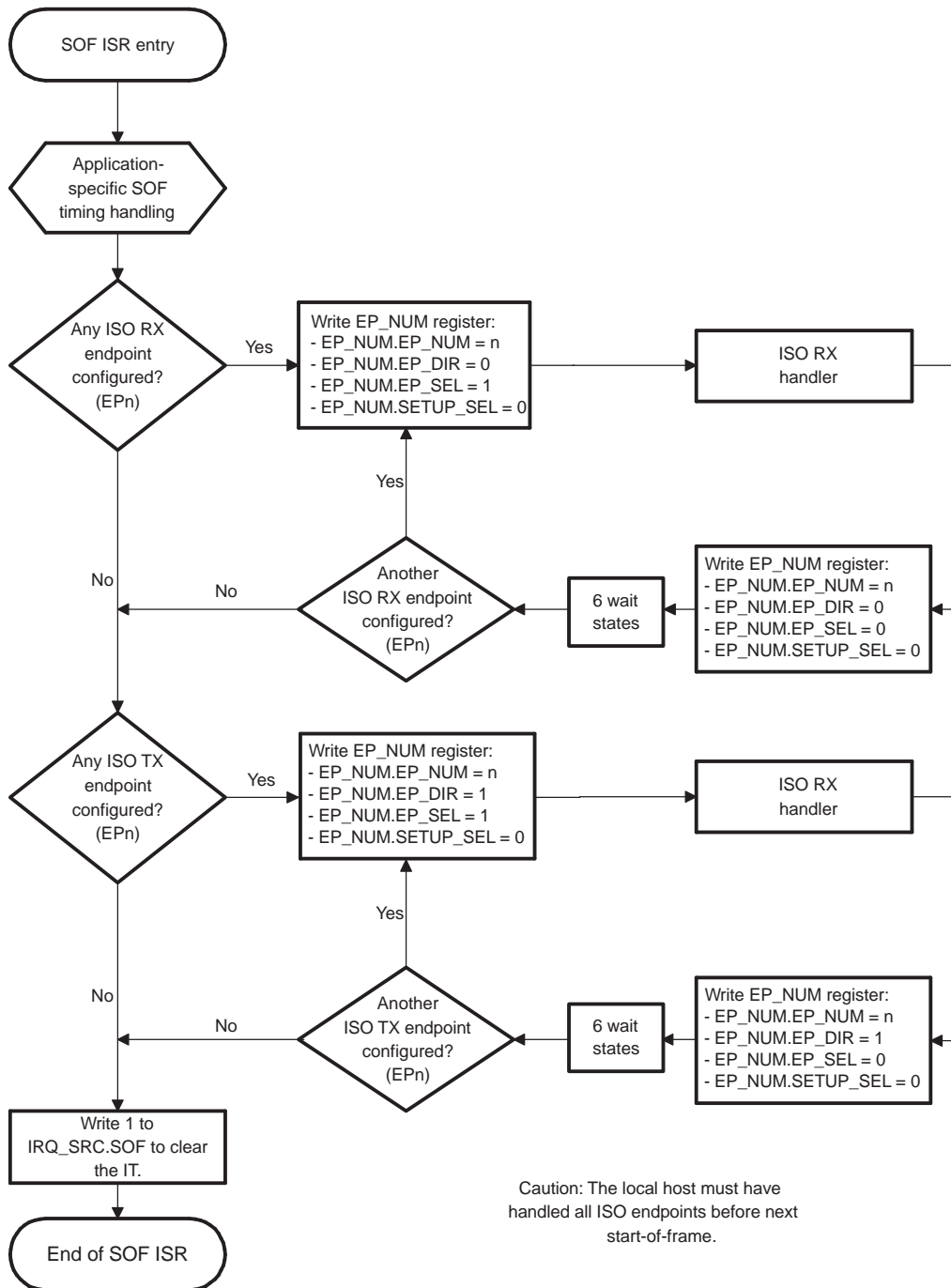


**Figure 28-80. Write Non-Isochronous TX FIFO Data Flowchart**

### 28.3.25 SOF Interrupt Handler

SOF interrupts to the CPU (also known as isochronous interrupts on VIM channel # 68 by default) occur once per USB frame. The CPU must handle data traffic for the isochronous endpoints at each SOF interrupt. In addition, the SOF ISR can handle any application-specific activity related to the implicit timing of the SOF interrupt. [Figure 28-81](#) shows the SOF ISR flowchart. The read ISO RX FIFO data and write ISO TX FIFO data procedures are shown in [Figure 28-82](#) and [Figure 28-83](#), respectively.

Figure 28-81. SOF Interrupt Handler Flowchart



Caution: The local host must have handled all ISO endpoints before next start-of-frame.

Figure 28-82. Read Isochronous RX FIFO Data Flowchart

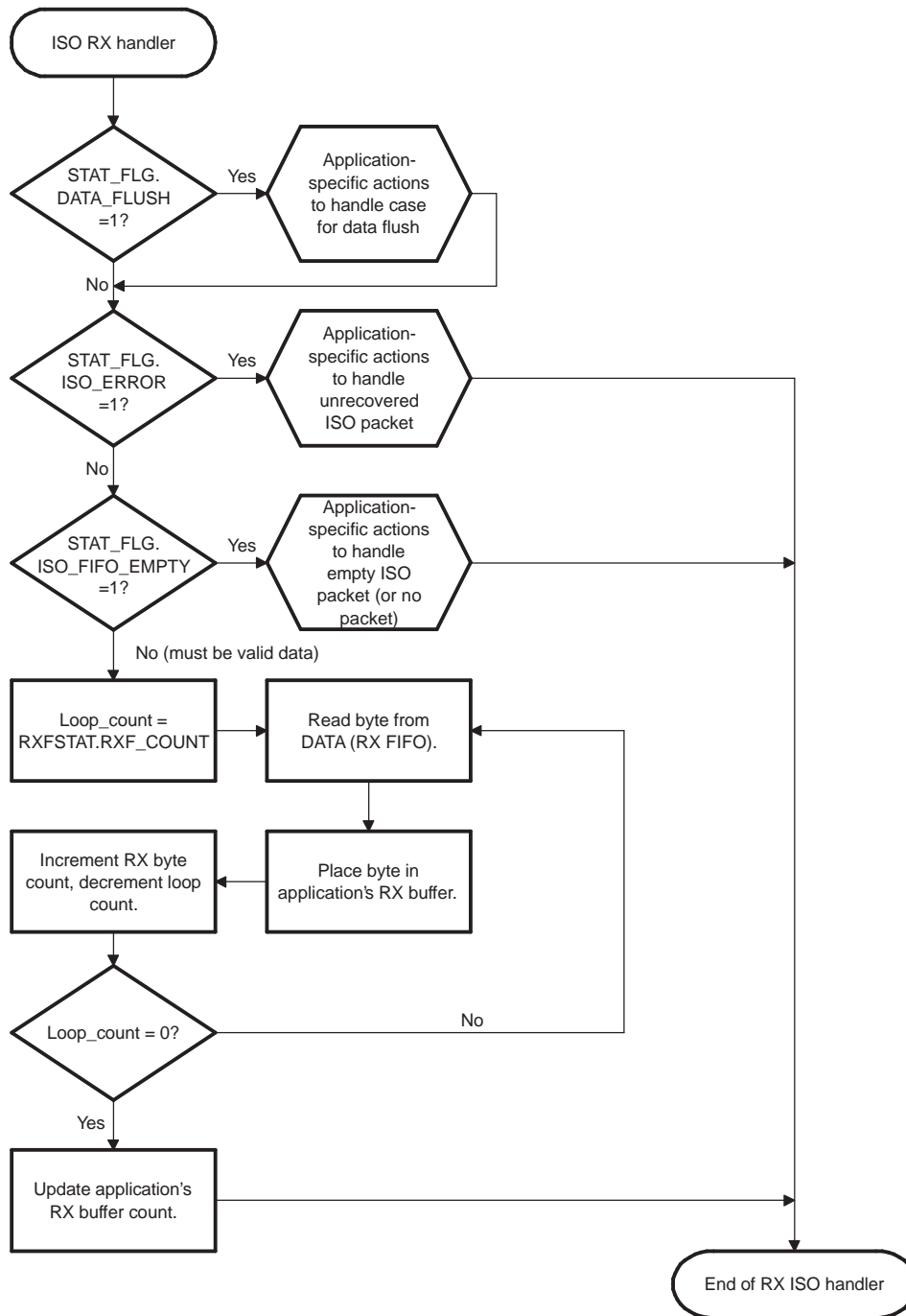
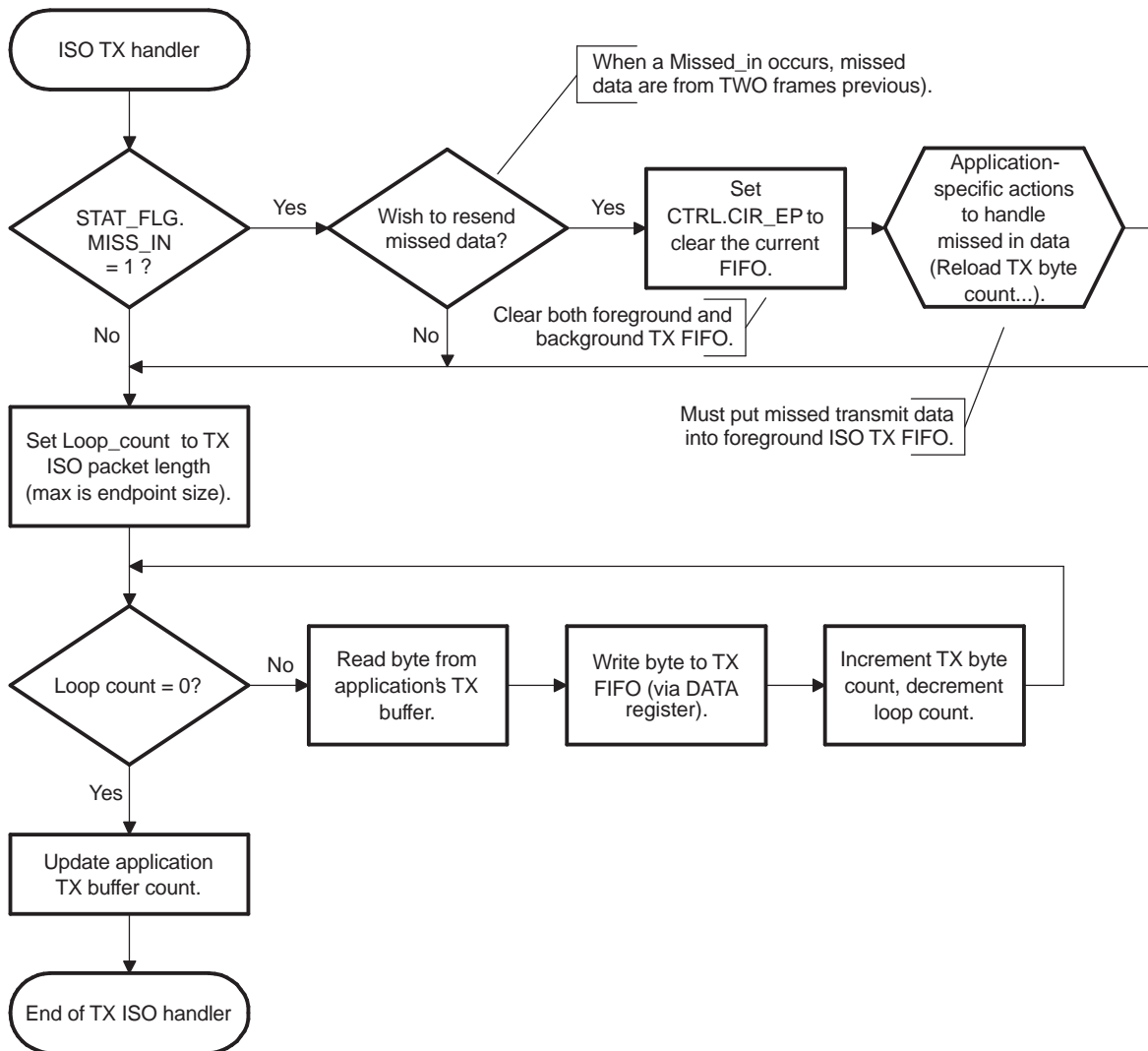


Figure 28-83. Write Isochronous TX FIFO Data Flowchart



### 28.3.26 Summary of USB Device Controller Interrupts

Table 28-56 lists the interrupt types by endpoint types.

Table 28-56. USB Device Controller Interrupt Type by Endpoint Type

Interrupt Type	General USB IRQs (VIM channel # 69 by default)			EP-Specific IRQs (Non-ISO Interrupt on VIM channel # 70 by default)		SOF ISO Interrupt on VIM channel # 68 by default) (Isochronous) SOF
	Setup (EP0)	Control (EP0) Out	Control (EP0) In	Bulk or InterruptOut	Bulk or InterruptIn	
Transaction ACKed		X	X	X	X	
Transaction NAKed (if enabled)		X	X	X	X	
Transaction Stalled		X	X	X	X	
Setup	X					
SOF						X

**Table 28-56. USB Device Controller Interrupt Type by Endpoint Type (continued)**

Interrupt Type	General USB IRQs (VIM channel # 69 by default)			EP-Specific IRQs (Non-ISO Interrupt on VIM channel # 70 by default)		SOF ISO Interrupt on VIM channel # 68 by default)	
	Setup (EP0)	Control (EP0) Out	Control (EP0) In	Other	Bulk or InterruptOut	Bulk or InterruptIn	(Isochronous) SOF
Device state changed				X			
RX DMA EOT (non_ISO)				X			
RX DMA trans count (non_ISO)				X			
TX DMA done (non_ISO)				X			

### 28.3.26.1 USB Device Controller Clock Control

The device global clock module (GCM) provides a single 48-MHz clock to the USB device controller and USB host controller. This is the VCLKA3\_S clock domain. This clock domain can be stopped by software to reduce USB host and USB device controller power consumption when USB functionality is not needed. The GCM controls the 48-MHz clock to the USB device controller via the following memory-mapped registers:

- SYS.CDDIS
- SYS.CDDISSET
- SYS.CDDISCLR

### 28.3.26.2 USB Device Controller Hardware Reset

Reset of the USB device controller is provided by the SYS module. The system reset signal is used to reset the USB host controller and the USB device controller. When held in reset, the USB device controller does not recognize any USB activity, and its registers have no effect on USB functionality.

### 28.3.27 DMA Operation

The USB device controller provides support for six DMA channels. Three receive DMA channels are reserved to OUT transfers (ISO or non-ISO) and three transmit DMA channels are reserved to IN transfers (ISO or non-ISO). It is not possible to operate DMA transactions on control EP0.

---

**NOTE:** The CPU must not access an endpoint used in a DMA transfer through the EP\_NUM, CTRL, and STAT\_FLG registers (in DMA, this remark applies after the CPU has set the CTRL.SET\_FIFO\_EN bit to enable the RX DMA transfer). In particular, the CPU must not set the halt feature while the endpoint is selected in the RXDMA\_CFG register.

---



---

**NOTE:** To use the DMA channels properly, you must set the DMA configuration during the address state interrupt (DS\_CHANGE).

---

The parameters used for DMA transactions (FIFO size, ISO endpoint, double-buffering, and pointers) are those defined for the associated endpoint.

### 28.3.27.1 Receive DMA Channels Overview

Receive DMA channels are programmed via the three RXDMA control registers. Each channel is assigned to a given endpoint number by assigning a non-zero value in RXDMA\_CFG.RXDMA<sub>n</sub>\_EP fields (a 0 value means the DMA channel is deselected). Received OUT data must be read when an RX DMA request is active, through the register DATA\_DMA. The RX FIFO accessed is that of the endpoint for which the DMA request is active (only one RX DMA request is active at a time).

The USB device controller receive DMA channels 0, 1, and 2 are connected to the device System DMA controller requests # 15, 19, and 29, respectively.

### 28.3.27.2 Non-Isochronous OUT (USB HOST -> CPU) DMA Transactions

During non-ISO transfers to a DMA operated OUT endpoint, a request to the CPU DMA controller is generated when data have been placed into endpoint FIFO and must be read. ACK and NAK interrupts are always disabled automatically by the core for DMA operated endpoints.

There are two dedicated maskable interrupts per DMA channel to control non-ISO OUT transfers.

### 28.3.27.3 End of Transfer Interrupt (IRQ\_SRC.RX<sub>n</sub>\_EOT)

This interrupt signals that the core has detected an end-of-transfer (EOT). EOT occurs in the two following cases:

- When the last valid transaction to the endpoint is either an empty packet (ACK and buffer empty) or a packet whose size is less than the physical endpoint buffer size (ACK and buffer not full)
- When the number of received transactions has reached the programmed value in the RXDMA<sub>n</sub>.RX<sub>n</sub>\_TC field, if the RXDMA<sub>n</sub>.RX<sub>n</sub>\_STOP bit has been set by the CPU

After an end of transfer interrupt, the CPU must set CTRL.SET\_FIFO\_EN for the endpoint to reenable the channel.

The CPU must not initiate a new RX DMA transfer until it receives an end-of-transfer interrupt.

### 28.3.27.4 Transaction Count Interrupt (IRQ\_SRC.RX<sub>n</sub>\_CNT)

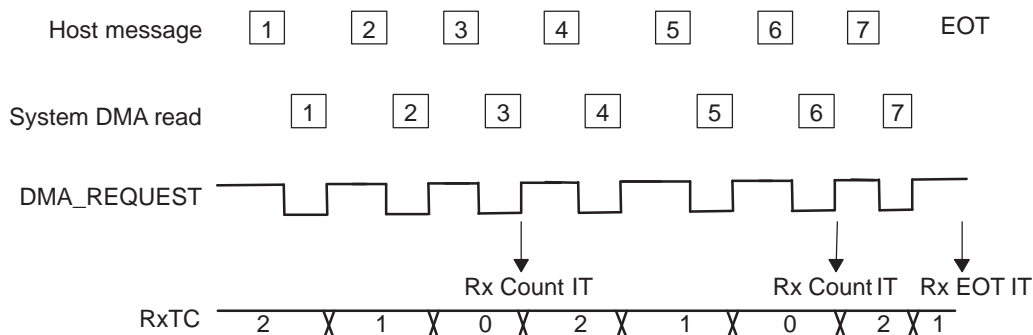
The intent of this interrupt is watermark control. It can be used by the CPU to monitor the file size of incoming transfers and take appropriate actions if, for instance, the file being received exceeds an expected size.

A transaction count interrupt does not disable the ongoing DMA transfer.

A transaction count interrupt occurs each time the number of received transactions (and not bytes) has reached the programmed value in the receive transaction counter for the DMA channel. One transaction has a size equal to the buffer size of the selected non-ISO endpoint. RX<sub>n</sub>\_COUNT interrupt is asserted even if RXDMA<sub>n</sub>.RX<sub>n</sub>\_STOP has been set; in that case, both RX<sub>n</sub>\_COUNT and RX<sub>n</sub>\_EOT interrupts are asserted (see [Figure 28-84](#) through [Figure 28-87](#)).

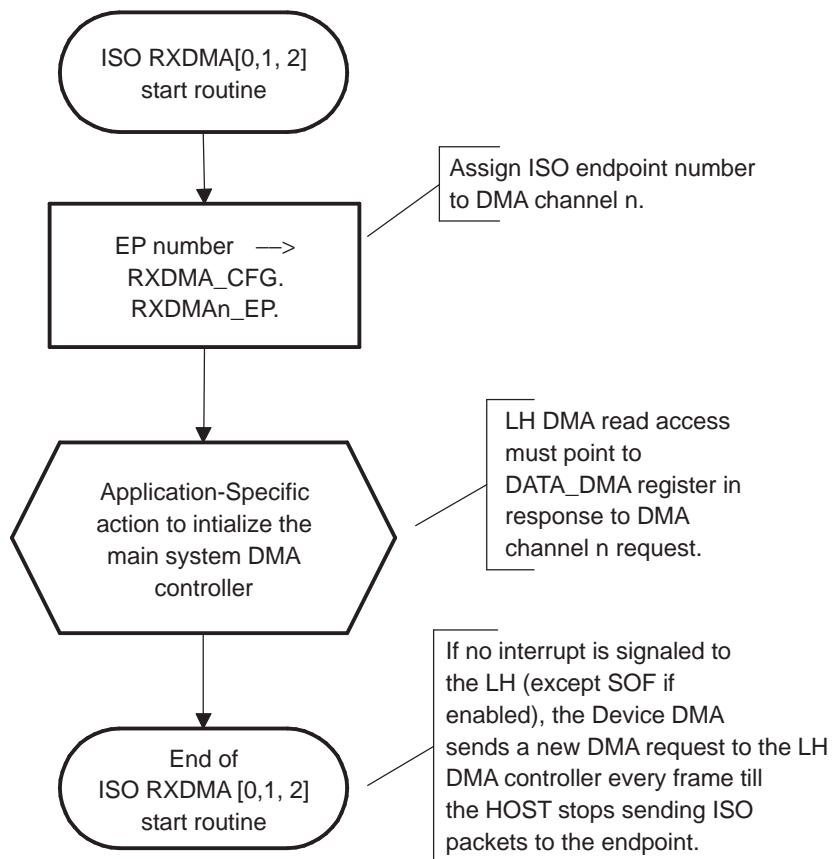
The transaction count watermark is programmed in the RXDMA<sub>n</sub> register.

**Figure 28-84. Non-ISO RX DMA Transaction Example (RX\_TC=2)**

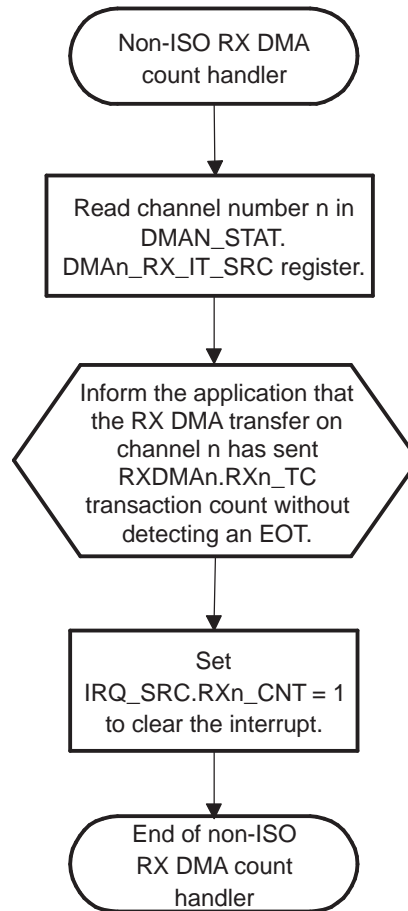




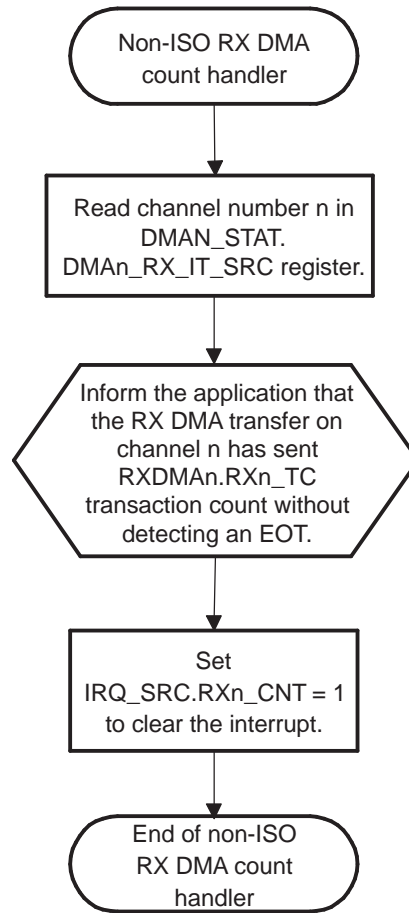
**Figure 28-85. Non-ISO RX DMA Start Routine**



**Figure 28-86. Non-ISO RX DMA EOT Interrupt Handler**



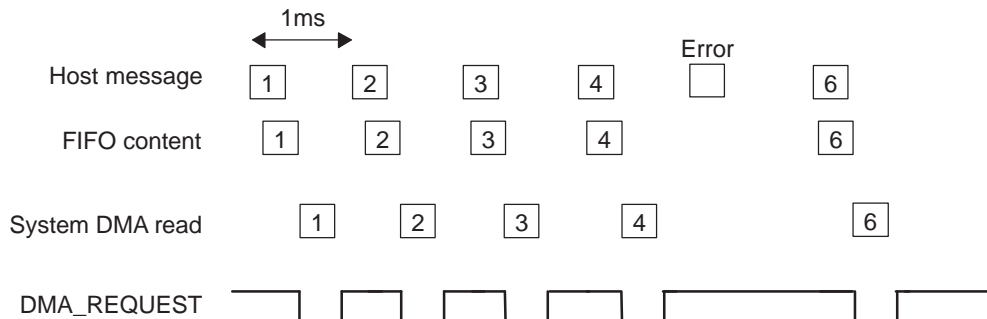
**Figure 28-87. Non-ISO RX DMA Transactions Count Interrupt Handler**



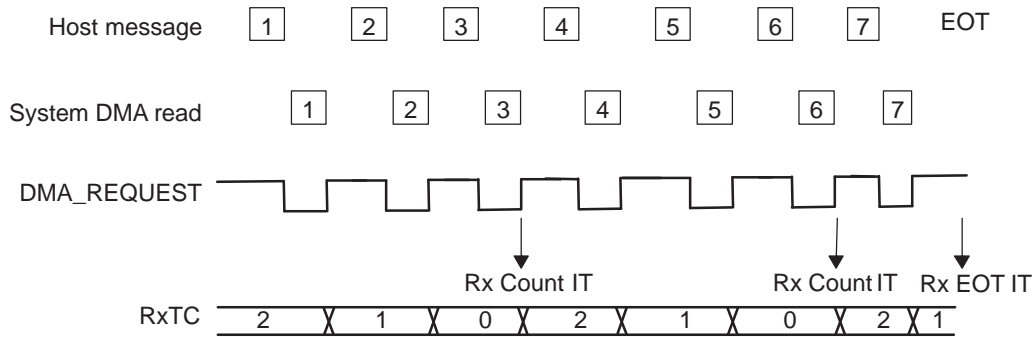
**28.3.27.5 Isochronous OUT (USB HOST -> CPU) DMA Transactions**

During ISO transfers to a DMA operated OUT endpoint, a request to the CPU DMA controller is generated every 1-ms frame when an isochronous data packet is received with no error. There is no interrupt associated with DMA transfer to ISO OUT endpoints (see [Figure 28-88](#) and [Figure 28-89](#)).

**Figure 28-88. ISO RX DMA Transaction**



**Figure 28-89. ISO RX DMA Start Routine**



**28.3.27.6 Transmit DMA Channels Overview**

Transmit DMA channels are programmed via the three TXDMA control registers. Each channel can be assigned to a given endpoint number by assigning a non-zero value in TXDMA\_CFG.TXDMA<sub>n</sub>\_EP (a 0 value means the DMA channel is deselected). The other three control registers (TXDMA0, TXDMA1, and TXDMA2) operate in a different manner for ISO and non-ISO endpoints. Transmitted data must be written into the DATA\_DMA when a TX DMA request is active. They are written into the TX FIFO of the endpoint associated with active request (only one TX DMA request active at a given time).

The USB device controller transmit DMA channels 0, 1, and 2 are connected to the device system DMA controller requests # 14, 20, and 28, respectively.

**28.3.27.7 Non-Isochronous IN (CPU -> USB HOST) DMA Transactions**

Non-ISO (bulk) TX DMA file transfers are virtually unlimited in size. The flowcharts depicted in [Figure 28-90](#) and [Figure 28-91](#) show how to handle small, medium, or large file transfers.

TXDMA0, TXDMA1, and TXDMA2 registers operate for non-ISO endpoints in the following manner. The transfer size counter (TXDMA<sub>n</sub>.TXN\_TSC) corresponds to either the number of bytes to transmit (EOT bit set) or the number of buffers to transmit (EOT bit cleared). The buffer size corresponds to the programmed size of the TX endpoint.

A request to the CPU main DMA controller is generated when the endpoint buffer is empty initially after that the START bit is set and then each time there is space free in TX FIFO for other TX packet to be written, until TXDMA<sub>n</sub>.TX<sub>n</sub>\_TSC counts down to zero. The request is removed when the buffer is full or when there are no more bytes of data to be sent.

A DMA transmit transfer done interrupt is signaled to the CPU after the last IN transaction completes successfully. This is after START bit was set and after TXDMA<sub>n</sub>.TX<sub>n</sub>\_TSC equals 0 for the selected DMA channel.

The CPU must not initiate a new TX DMA transfer until it receives a TX\_DONE interrupt.

A small file transfer less than 1024 bytes can be achieved in a single pass DMA signaled by a single interrupt completion. A file size equal to or greater than 1024 bytes needs two or more DMA passes, signaled by an interrupt completion after each pass.

[Figure 28-90](#) shows the necessary steps to prepare and permit a TX DMA transfer of any size. It also effectively starts the initial DMA transfer. The completion of this DMA task is signaled to the CPU via a DONE interrupt, whose handler is shown in [Figure 28-91](#). The start routine and the associated interrupt handler are tightly coupled.

**Figure 28-90. Non-ISO TX DMA Start Routine**

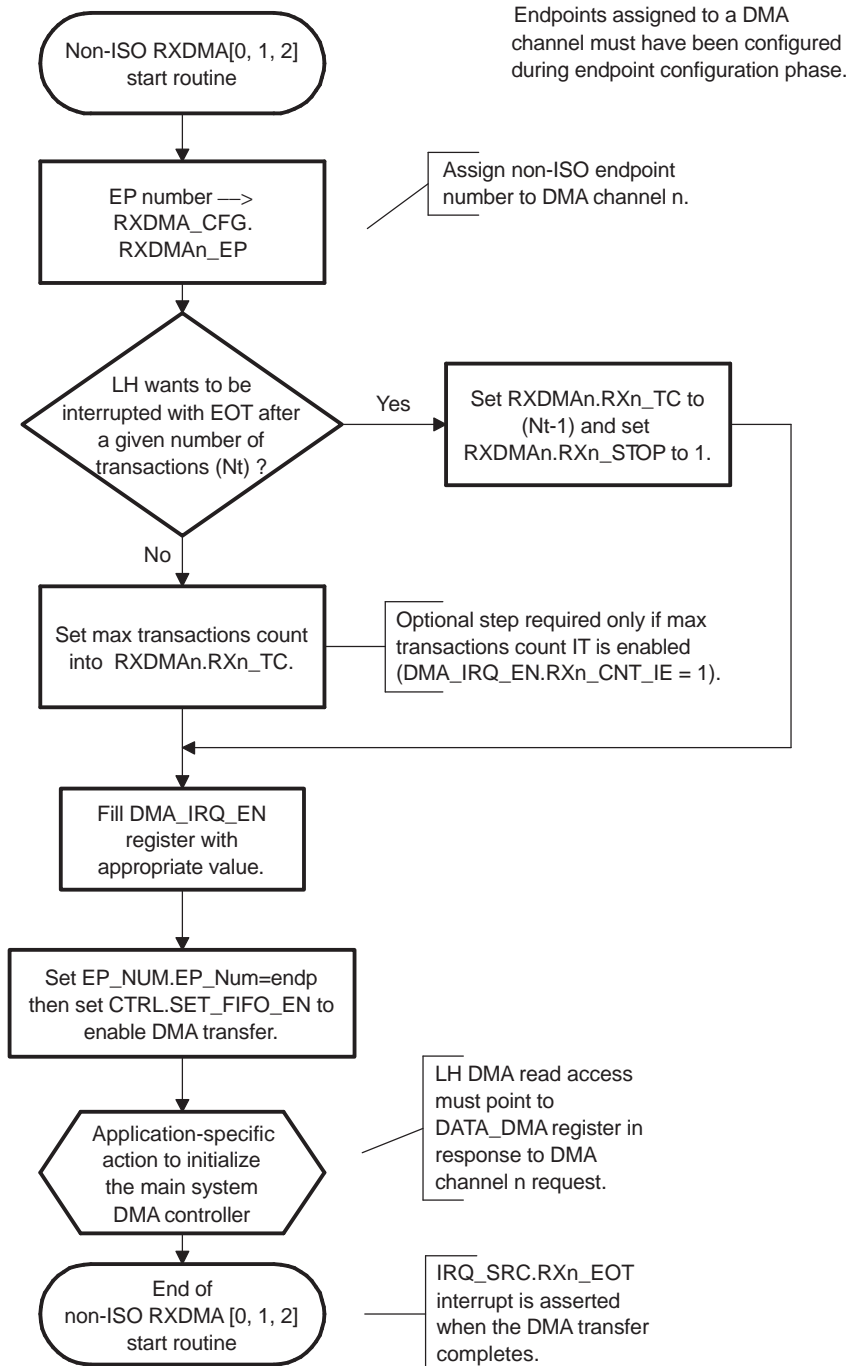
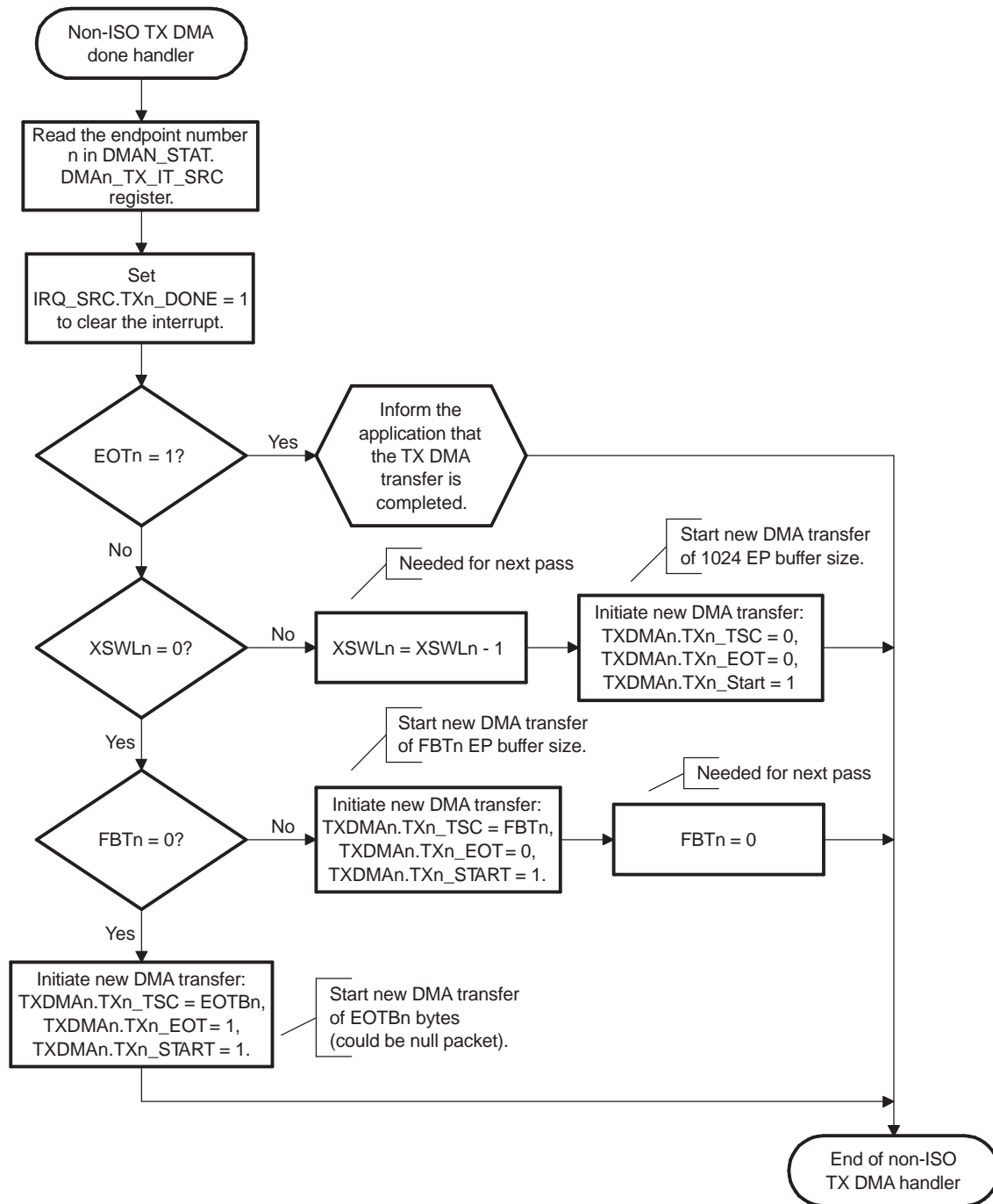


Figure 28-91. Non-ISO TX DMA Done Interrupt Handler



**Example 28-1. 100603 Bytes to Transfer via 32 Bytes IN Bulk Endpoint**

This gives XSWL=0x3, FBT=0x47, EOTB=0x1b,

which means five passes of DMA transfer, signaled by 5 TXn\_DONE interrupts, are required:

1. EOT=0, FBT=0, loop=3 32768 bytes transferred (1024 x 32 bytes)
2. EOT=0, FBT=0, loop=2 32768 bytes
3. EOT=0, FBT=0, loop=1 32768 bytes

**Example 28-1. 100603 Bytes to Transfer via 32 Bytes IN Bulk Endpoint (continued)**

4. EOT=0, FBT=0x47, loop=0 2272 bytes (71 x 32 bytes)
5. EOT=1, FBT=0x1B, loop=0 27 bytes

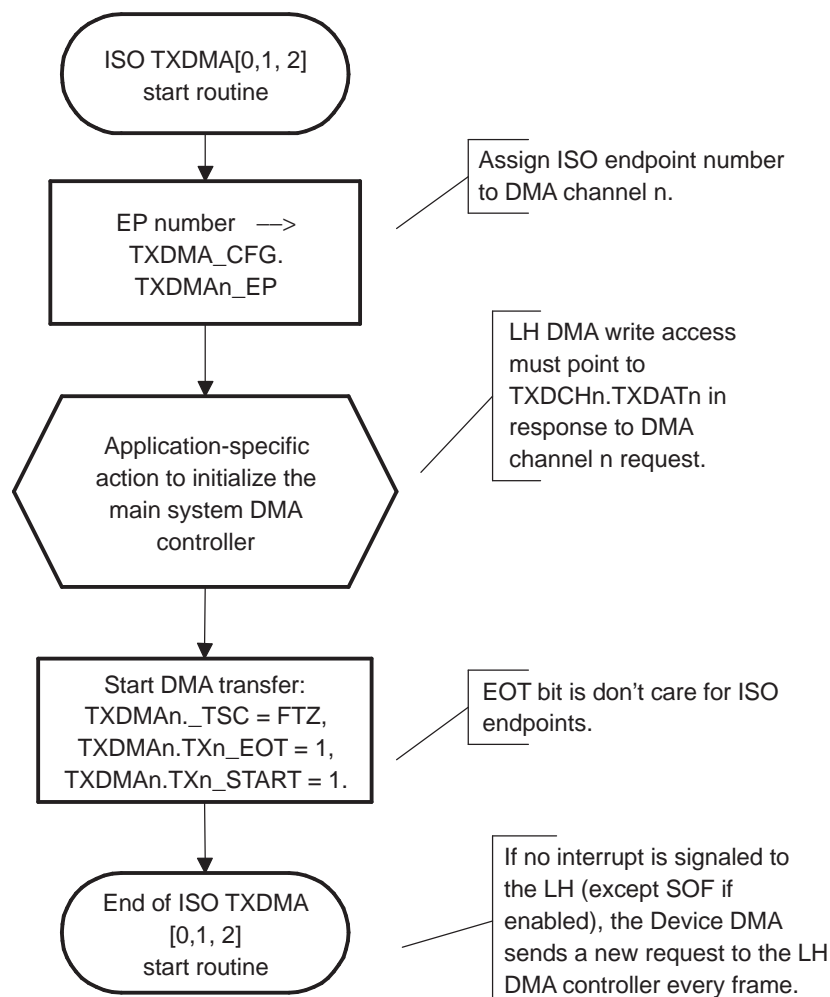
**28.3.27.8 Isochronous IN (USB HOST -> CPU) DMA Transactions**

For ISO endpoints, the transfer size counter (TXDMA<sub>n</sub>.TX<sub>n</sub>\_TSC) corresponds to the number of bytes to transmit. The programmed size must not exceed the programmed buffer size of the endpoint. Otherwise, the result is unpredictable (see [Figure 28-92](#)).

A request to the CPU main DMA controller is generated when the endpoint buffer is empty initially after the START bit is set, and then after each SOF (every 1 ms). The request is removed when the number of bytes written in the buffer matches the TXDMA<sub>n</sub>.TX<sub>n</sub>\_TSC value.

During ISO transfers to a DMA operated IN endpoint, a request to the CPU system DMA controller is generated every 1-ms frame when an isochronous data packet is received with no error. There is no special interrupt associated with the DMA transfer.

No interrupt is signaled to the CPU during DMA operation to ISO IN endpoints.

**Figure 28-92. ISO TX DMA Start Routine**

### 28.3.27.9 Important Note on DMA Requests

For each direction, only one DMA request can be active at any time. A request must then be serviced to allow the next pending request on the same direction to be asserted. In particular, a TX DMA request is asserted at each start-of-frame if a TX DMA channel is configured for an isochronous endpoint; this request must be serviced imperatively.

### 28.3.27.10 Note on DMA Channels Deconfiguration

It is recommended that the CPU wait for an EOT (RX) or a DONE (TX) interrupt before disabling the channel by writing a value 0 in the TX/RXDMA\_CFG register. However, if needed by the application, the CPU can unselect the endpoint number in the TX/RXDMA\_CFG register during a DMA transfer. The resulting behavior is:

- For RX transfer:
  - If RX DMA request is active for the endpoint when the endpoint is unselected, deconfiguration is effective only at the end of the RX DMA request (that is, after all the DMA data have been read). When double-buffering is used, the deconfiguration is effective after both buffers have been read (if both buffers were not empty at unselection). An EOT interrupt is asserted if an end-of-transfer is detected.
  - If the RX DMA request is not active when unselection occurs, the effect is immediate.
- For TX transfer:
  - If the request is active when the endpoint is unselected, deconfiguration is effective after the TX DMA request has been handled and the TX data have been sent through an IN transaction (both buffers in case of double-buffering with both buffers full). No TX\_DONE interrupt is asserted even if TXDMA<sub>n</sub>.TSC bit value is 0 after the transaction.
  - If the TX DMA request is inactive when the endpoint is unselected, deconfiguration is effective when all data available in TX buffer(s) have been sent through IN transaction(s). If the TXDMA<sub>n</sub>.TSC value is 0 at this point, no TX\_DONE interrupt is asserted. If TX\_DONE interrupt had already been asserted when the endpoint is deselected, configuration is effective only after the TX\_DONE interrupt handling.

TX/RXDMA\_CFG.TX/RXDMA<sub>n</sub>\_EP reflects the endpoint value until deconfiguration is effective. The CPU must read this register to know if the channel has been disabled yet or not. It must wait until the read value is 0 before performing other actions to the endpoint. After effective deconfiguration, all transactions to the endpoint generate an endpoint-specific interrupt (if non-transparent).

If the selected endpoint is of isochronous type, deconfiguration is effective after the TX/RX request has been serviced, and the subsequent isochronous transactions are handled at SOF interrupt through the endpoint registers (EP\_NUM and STAT\_FLG).

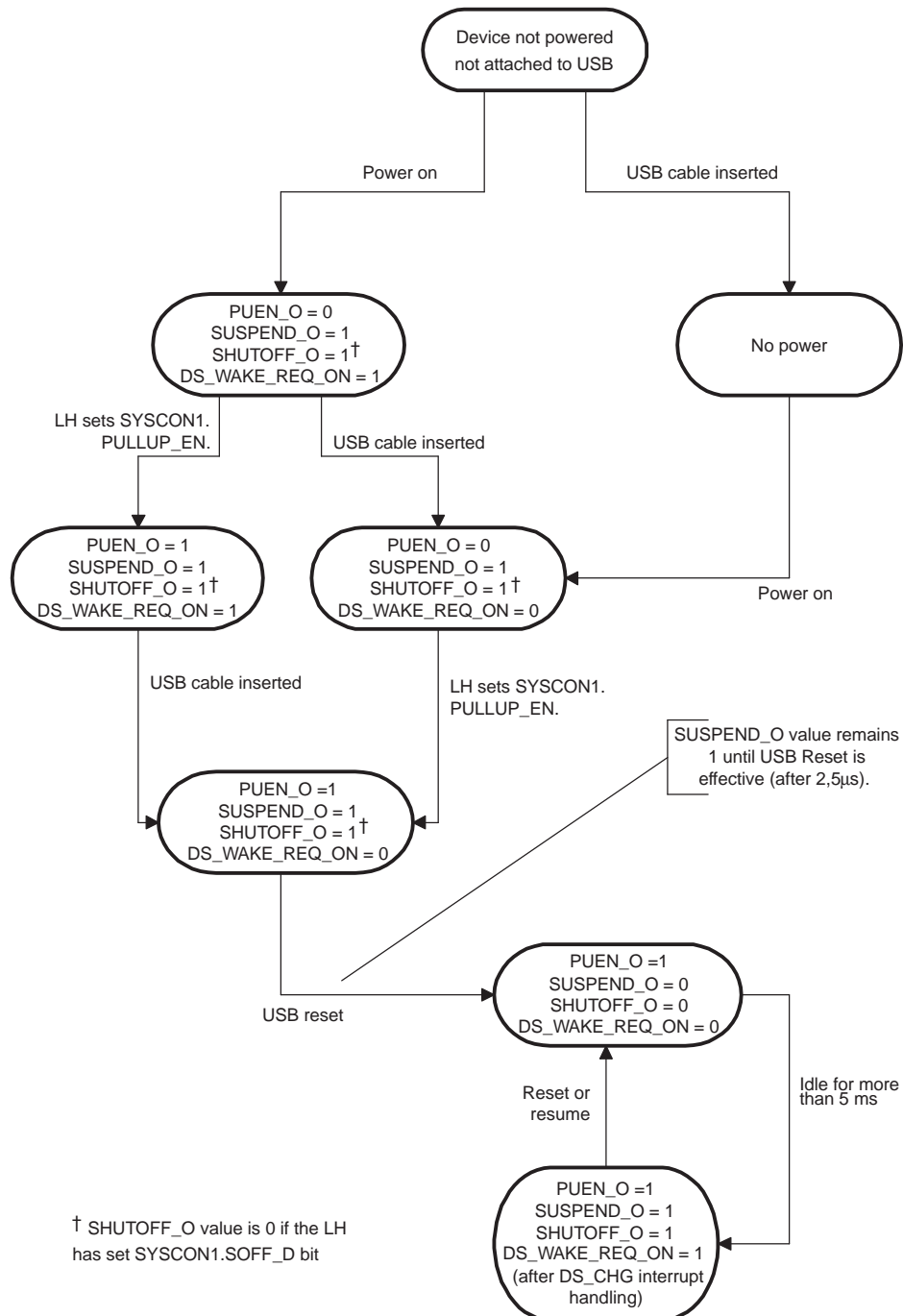
## 28.3.28 Power Management

Figure 28-93 shows the values assigned to the USB device controller signals concerned with power management, in the function of the device state. These signals are:

- PUEN\_O: Pullup enable signal, always reflecting the SYSCON1.PULLUP\_EN register bit
- SHUTOFF\_O: Power circuitry shutoff signal, controlled by the core and the SYSCON1.SOFF\_DIS bit
- DS\_WAKE\_REQ\_ON: Deep-sleep wake request, asserted low when the interface clock is needed
- SUSPEND\_O: Suspend signal, asserted high when the device is in suspend mode



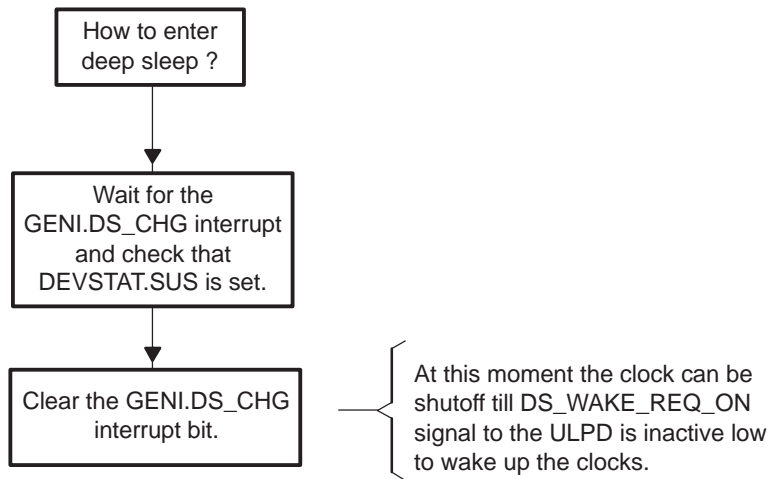
**Figure 28-93. Power Management Signal Values**



A SHUTOFF\_O value is 0 is the LH has set SYSCON1.SOFF\_D bit.

From a software point of view, [Figure 28-94](#) shows the reaction. This flowchart does not need to be implemented; it only reflects the way the module can enter deep sleep.

**Figure 28-94. Power Management Flowchart**



## 28.4 USB Connectivity

This section includes:

- Transceiver interface
- Selecting and configuring USB connectivity
- Transceiver signaling
- Host controller connectivity with USB transceivers
- USB function controller connectivity with USB transceivers
- USB hardware considerations

### 28.4.1 Transceiver Interface

The device provides three USB ports, two of which may be used simultaneously. In a USB system, a USB transceiver is needed for each USB port. It converts between signaling appropriate for the USB controllers and signaling appropriate for the USB cable. This device does not provide an integrated USB transceiver.

Several different types of external transceiver signaling are supported. Signaling between the USB controller and the external USB transceiver for monitoring and controlling the differential USB signal is done via a 6-wire signaling interface, with two or more additional control signals provided either by additional signals or via an I2C link.

### 28.4.2 Selecting and Configuring USB Connectivity

The process of selecting desired USB connectivity and configuring the device for that connectivity can be done using the steps listed below.

#### 28.4.2.1 Choose the Desired USB Functionality

Choose the desired USB functionality. The USB module on this microcontroller supports the following configurations:

- 1 USB Host Port + 1 USB Device Port, OR
- 2 USB Host Ports

Interface signals for the USB host port #2 are multiplexed with interface signals for the USB device port. If the application requires the use of the USB device port, then the USB host port #2 must first be powered down.

#### 28.4.2.2 Select How USB Functionality Is Multiplexed to Pins

Refer to [Table 28-1](#) for information on multiplexing and control of USB host ports and USB device port interface signals at the top level.

### 28.4.3 Transceiver Signaling

The USB module on this device uses a unidirectional (6-wire) signaling type for the transceiver. The signaling is shown in [Table 28-57](#).

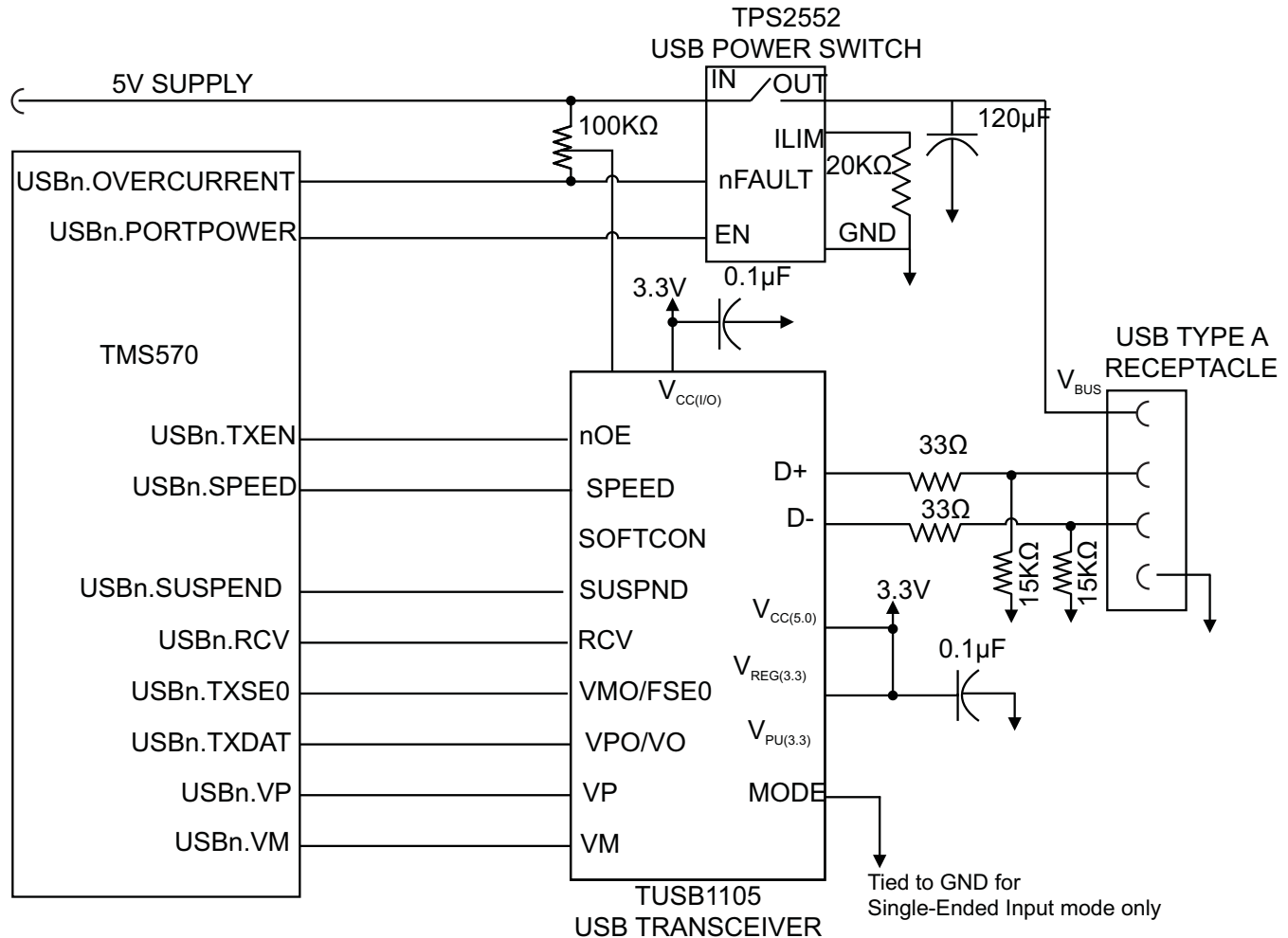
**Table 28-57. Signaling Between USB Controller and Unidirectional USB Transceiver Using DAT/SE0 Signaling**

Logical Signal Name(s)	Pin Direction	Transceiver Pin Direction	Description																									
nOE	Output	Input	When low, USB transceiver drives D+ and D-.																									
DAT and SE0	Output	Input	Controls the values output by the USB transceiver on D+ and D- when nOE is low. Ignored when nOE is high.																									
			<table border="1"> <thead> <tr> <th>nOE</th> <th>DAT</th> <th>SE0</th> <th>D+</th> <th>D-</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td></td> <td>1</td> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td></td> <td>X</td> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>X</td> <td>X</td> <td>Undriven</td> <td>Undriven</td> </tr> </tbody> </table>	nOE	DAT	SE0	D+	D-	0	0	0	0	1		1	0	1	0		X	1	0	0	1	X	X	Undriven	Undriven
nOE	DAT	SE0	D+	D-																								
0	0	0	0	1																								
	1	0	1	0																								
	X	1	0	0																								
1	X	X	Undriven	Undriven																								
RCV	Input	Output	Output from transceiver differential receiver.																									
			<table border="1"> <thead> <tr> <th>D+</th> <th>D-</th> <th>RCV</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>X</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>X</td> </tr> </tbody> </table>	D+	D-	RCV	0	0	X	0	1	0	1	0	1	1	1	X										
D+	D-	RCV																										
0	0	X																										
0	1	0																										
1	0	1																										
1	1	X																										
RCVVP	Input	Output	Output from transceiver single-ended D+ signal receiver .																									
			<table border="1"> <thead> <tr> <th>D+</th> <th>RCVVP</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> </tr> </tbody> </table>	D+	RCVVP	0	0	1	1																			
D+	RCVVP																											
0	0																											
1	1																											
RCVVM	Input	Output	Output from transceiver single-ended D- signal receiver.																									
			<table border="1"> <thead> <tr> <th>D-</th> <th>RCVVM</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> </tr> </tbody> </table>	D-	RCVVM	0	0	1	1																			
D-	RCVVM																											
0	0																											
1	1																											
SPEED	Output	Input	Determines transceiver D+, D- output characteristics. 0 = Low speed. 1 = Full speed. Transceivers with I <sup>2</sup> C interfaces can implement this functionality as a control bit rather than a pin.																									
SUSPEND	Output	Input	Controls transceiver power down modes. 0 = Active mode. 1 = Low-power mode. Transceivers with I <sup>2</sup> C interfaces can implement this functionality as a control bit rather than a pin.																									

This device does not support unidirectional transceivers that implement VPO/VMO signaling.

#### 28.4.4 Host Controller Connectivity With USB Transceivers

To provide a robust USB solution, a system that provides a USB host controller must implement certain features. These features include a type-A receptacle, power on the VBUS signal (can be switched or unswitched power), transient suppression, pulldown resistors, and USB-compatible downstream port transceiver.

**Figure 28-95. Pin Group 0 USB Host Using 6-Wire USB Transceiver**

Proper initialization of the device to support this mode of operation requires proper setting of the top-level pin multiplexing (see [Section 28.4.2.2](#)).

### 28.4.5 USB Function Controller Connectivity With USB Transceivers

To provide a robust USB solution, a system that provides a USB device controller port must implement certain features. These features include:

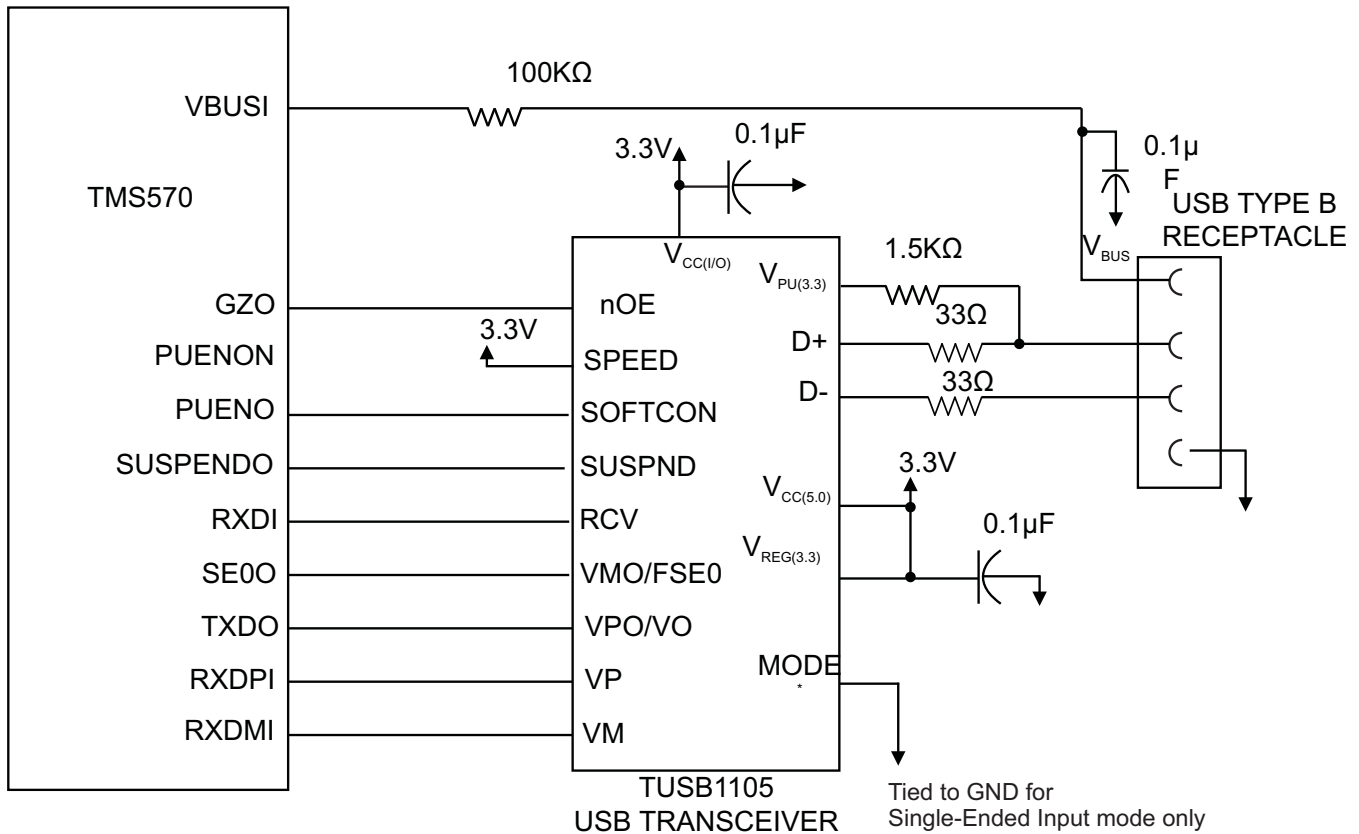
- A USB type-B or mini-B receptacle
- VBUS power detection
- Transient suppression
- A controllable pullup resistor to the D+ or D- line
- USB-compatible upstream port transceiver

Optional weak pulldown resistors can be used to hold the D+ and D- signals at voltages below the USB transceiver V<sub>I/L</sub> level when there is no USB host connected to the USB Type B connector. By keeping D+ and D- voltages below V<sub>I/L</sub>, the USB transceiver I<sub>DDQ</sub> can be reduced. Choice of value for the pulldown resistors must be made carefully so that the circuit meets the requirements of the *USB Specification*.

The USB function controller only supports implementation as a full-speed USB device. As such, the pullup resistor must be connected to the D+ signal to indicate implementation of a full-speed USB device.

This device supports a 6-wire connection to the USB transceiver.

Figure 28-96. USB Device Connections on USB Pin Group 0 Using 6-Wire Transceiver



Proper initialization of the device to support this mode of operation requires proper setting of the top-level pin multiplexing (see [Section 28.4.2.2](#)).

## 28.4.6 USB Hardware Considerations

### 28.4.6.1 VBUS Power Switching for USB Type A Host Receptacles

The USB specification places several VBUS requirements on USB hosts, including current capability, droop, and other important characteristics. Circuits that meet the USB specification requirements can be implemented using Texas Instruments devices such as the TPS2552 and TPS2553 power distribution switch devices. Further information, including data sheets and application notes, can be found at the Texas Instruments web site.

### 28.4.6.2 Transient Suppression for USB Connectors

It is important to provide transient suppression for USB connectors. Electrostatic discharge that occurs when a user connects or disconnects a USB cable can have a dramatic effect on a system if not suppressed. Texas Instruments offers several devices for transient suppression on USB connections, such as the SN65220, SN65240, and SN75240 universal serial bus port transient suppressor devices. Further information, including data sheets and application notes, can be found at the Texas Instruments web site.

### 28.4.6.3 VBUS Monitoring for USB Function Controller

A USB function controller must be capable of monitoring the VBUS voltage provided by the upstream USB host controller. The device provides the input pin USB\_VBUS, which is provided to the USB function controller. This input is a CMOS input that is not rated for the full VBUS range defined by the USB specification. An external signal level shifter is required to convert the VBUS signal range to a range suitable for the USB\_VBUS pin.

#### 28.4.6.4 USB D+ Pullup Enable for USB Function Controller

The USB\_PUEN pin is driven high when the pullup is active and is driven low when the pullup is inactive. In this mode of operation, the pullup resistor can be connected directly between the USB\_PUEN and the D+ signal. Some transceivers such as the TUSB1105 from Texas Instruments have an additional input pin called SOFTCON. When this pin is driven High, the D+ gets pulled up through an external resistor to 3.3V. The USB\_PUEN can be connected to the SOFTCON pin in this case.

#### 28.4.6.5 USB D+, D- Pulldown for USB Function Controller

System implementations that use the USB function controller and are sensitive to supply current requirements can implement weak pulldown resistors on the USB D+ and D- signals associated with the USB Type-B receptacle. When there is no host controller attached upstream of the USB Type-B receptacle, the undriven D+ and D- wires can float to voltages that cause excessive current consumption by the USB transceiver. Weak pulldowns can help prevent this problem. Selection of pulldown resistors depends on transceiver characteristics, D+ pullup resistor implementation, and board layout, and must be designed to meet USB D+ and D- signal requirements.

## **Data Modification Module (DMM)**

---

---

---

This chapter describes the functionality of the Data Modification Module (DMM), which provides the capability to modify data in the entire 4 GB address space of the device from an external peripheral, with minimal interruption of the application.

<b>Topic</b>	<b>Page</b>
<b>29.1 Overview</b> .....	<b>1814</b>
<b>29.2 Module Operation</b> .....	<b>1815</b>
<b>29.3 Control Registers Summary</b> .....	<b>1820</b>



## 29.1 Overview

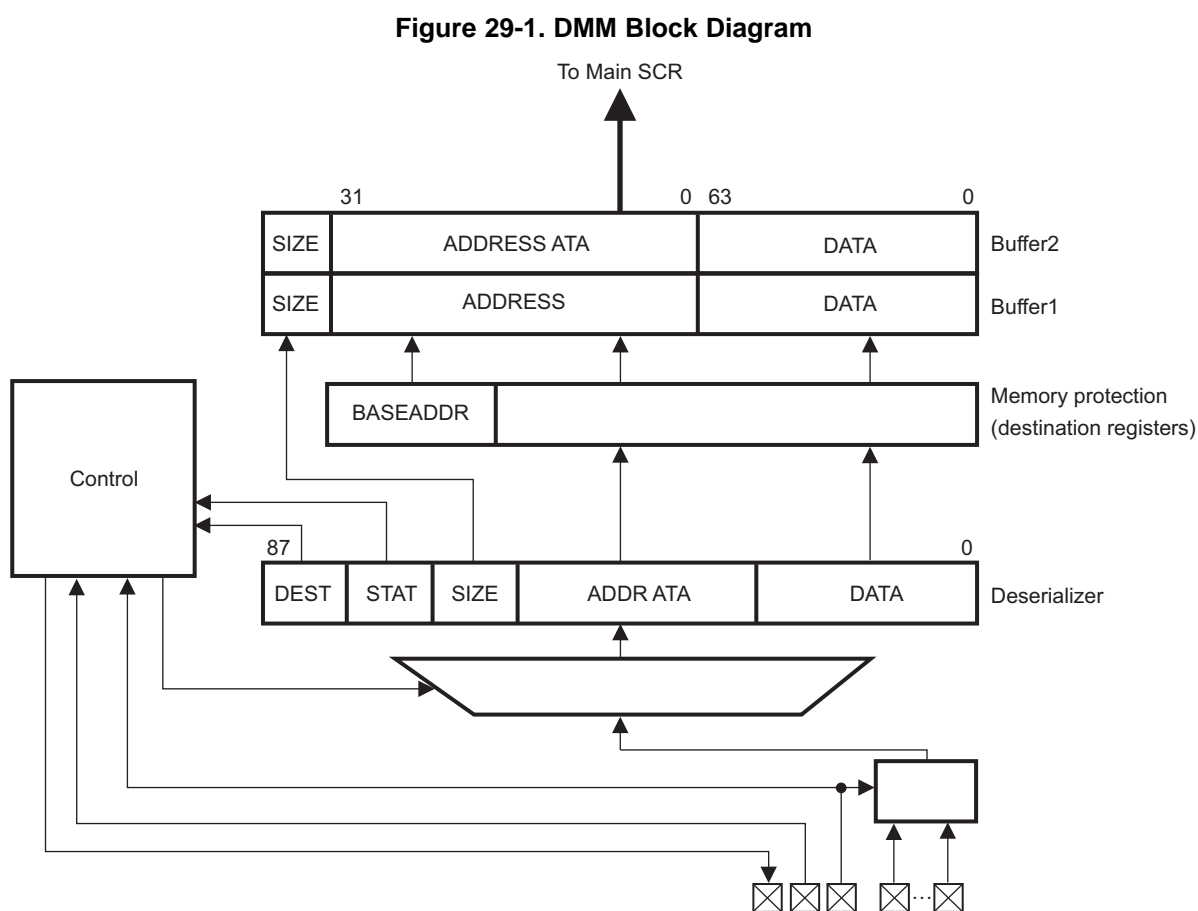
### 29.1.1 Features

The DMM module has the following features:

- Acts as a bus master, thus enabling direct writes to the 4GB address space without CPU intervention
- Writes to memory locations specified in the received packet (leverages packets defined by trace mode of the RAM trace port (RTP) module)
- Writes received data to consecutive addresses, which are specified by the DMM module (leverages packets defined by direct data mode of RTP module)
- Configurable port width (1, 2, 4, 8, 16 pins)
- Up to 100 Mbit/s pin data rate
- Unused pins configurable as GIO pins

### 29.1.2 Block Diagram

Figure 29-1 shows the block diagram for the DMM.



## 29.2 Module Operation

The DMM receives data over the DMM pins from external systems and writes the received data directly to the base address programmed in the module plus offset address given in the packet or into a buffer specified by start address and length. It leverages the protocol defined by the RAM Trace Port (RTP) module to have a common interface definition for external systems. It can also be used to connect an RTP and DMM module together for fast processor intercommunication.

The DMM module provides two modes of operation:

- **Trace Mode:** In this mode, the DMM writes the received data directly to an address which is calculated from the base address programmed into the destination register (Section 29.3.12; Section 29.3.14) plus the offset address contained in the received packet. An interrupt can be generated when data is written the lowest address of a programmed region. This capability enables the sender to raise an interrupt at the receiver while sending specific information.
- **Direct Data Mode:** In this mode, the DMM writes the received data into an address range of the 4GB address space. The buffer start address (Section 29.3.8) and blocksize (Section 29.3.9) is programmable in the DMM module. When the buffer reaches its end address, the buffer pointer wraps around and points to the beginning of the buffer again. The EO\_BUFF flag (Section 29.3.5) will be set and if enabled, an interrupt will be generated to indicate a buffer-full condition. Another interrupt, can be configured to indicate different buffer fill levels. This can be accomplished by programming a certain fill level into the DMMINTPT register (Section 29.3.11). The PROG\_BUFF flag (Section 29.3.5) indicates that this level has been reached.

Data will be captured by the input buffer and moved to the appropriate bit field in the deserializer. When the deserializer is completely full, the data will be moved to the output buffer register. A two-level buffer is implemented to avoid overflow conditions if the internal bus is occupied by other transactions. In addition the DMMENA signal can be used to signal the external hardware that an overflow might occur if more data is sent. The automatic generation of the DMMENA signal can be configured by setting the ENAFUNC bit (Section 29.3.16). While the DMMENA signal is active, the DMM module will not receive any new data.

The DMM is a bus master and forwards the received data to the bus system. The write operation will be minimally intrusive to the program flow, because the CPU/DMA access will only be blocked if the CPU/DMA accesses the same resource as the DMM.

To prevent an external system from overwriting critical data in the memory while configured in Trace Mode, a memory protection mechanism is implemented via a programmable start address and block size of a region. A maximum of four destinations with two regions each are supported.

For proper operation, at least DMMCLK, DMMSYNC and DMMDATA[0] need to be programmed in functional mode (Section 29.3.16). If a large amount of data should be transmitted in a short time, more data pins should be used in functional mode. The module supports 1, 2, 4, 8, or 16-pin configurations.

The module can be configured to handle a free running clock provided on DMMCLK (Section 29.3.1). Clock pulses between two DMMSYNC pulses which exceed the number of valid clock pulses for a packet will be ignored.

### 29.2.1 Data Format

Below is a description of the packet and frame format.

#### 29.2.1.1 Clocking Scheme

The DMM supports both continuous and noncontinuous clocking. The clock received on DMMCLK in the continuous clocking scheme is a free-running clock. In noncontinuous clocking scheme, the clock will stop after each packet and will start with the reception of a DMMSYNC signal.

#### 29.2.1.2 Trace Mode Packet

Figure 29-2 illustrates the trace mode packet format. One packet consists of 2 bits (DEST) denoting the destination in which the data is stored, 2 status bits (STAT), the 2-bit SIZE of the data, the 18-bit address of where the data should be written to, and a variable data field.

The DEST bits (Table 29-1) will be used to determine which destination register applies to the transmitted

data and the received address determines if the packet falls into a valid region of the destination area. If the address is valid, the base address, programmed in one of the destination registers (Section 29.3.12; Section 29.3.14) of this particular region will be applied to create the complete 32-bit address for the destination. The DMM module only takes action on a "11" setting of the STAT bits (Table 29-2). This signals that an overflow in the transmitting hardware module has occurred. If this is the case the SRC\_OVF flag (Section 29.3.5) will be set and the received data will be written to the address specified in the packet. The size information of the data transmitted in the packet is denoted in the SIZE bits (Table 29-3) of the packet. Depending on the SIZE information, the module expects to receive only this amount of data.

**Figure 29-2. Trace Mode Packet Format**

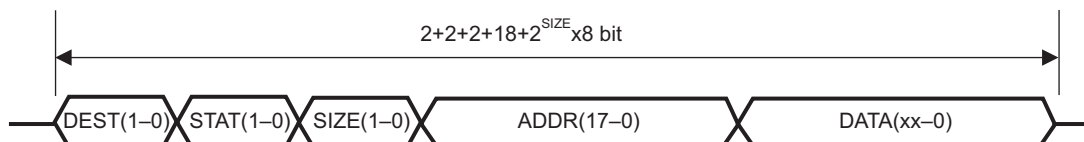


Table 29-1 through Table 29-3 illustrate the encoding of packet format in trace mode.

**Table 29-1. Encoding of Destination Bits in Trace Mode Packet Format**

DEST[1:0]	Destination
00	Dest 0
01	Dest 1
10	Dest 2
11	Dest 3

**Table 29-2. Encoding of Status Bits in Trace Mode Packet Format**

STAT[1:0]	Status
00	don't care
01	don't care
10	don't care
11	overflow

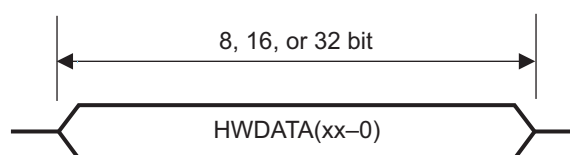
**Table 29-3. Encoding of Write Size in Packet Format**

SIZE[1:0]	Write Size
00	8 bit
01	16 bit
10	32 bit
11	64 bit

### 29.2.1.3 Direct Data Mode Packet

Figure 29-3 illustrates the direct data mode packet format.

**Figure 29-3. Direct Data Mode Packet Format**



The packet consists only of data bits and no header information. It can be 8-, 16- or 32-bit wide. A variable packet width is not supported because the DMM module will check the number of incoming bits (DMMCLK cycles) for error detection. The DMM will write the received data to the destination once the programmed number of bits has been received.

If the programmed word width does not correspond to the received data, the following actions will be taken.

- If the received data is greater than the programmed width, only the configured number of bits are transferred into the RAM buffer, the additional bits are discarded.
- If the received number of bits is smaller than the programmed width, no data will be written to the buffer, because a new DMMSYNC signal has been received before the expected number of bits.

### 29.2.2 Data Port

The packet will be received in several subpackets, depending on the width of the external data bus (DMMDATA[y:0]) and the amount of data to be transmitted. [Table 29-4](#) illustrates the number of clock cycles required for a complete packet.

**Table 29-4. Number of Clock Cycles per Packet**

Port Width/ Pins	Write Size in Bits			
	8	16	32	64
1	32	40	56	88
2	16	20	28	44
4	8	10	14	22
8	4	5	7	11
16	2	3	4	6

The user can program the port width in the DMMPC0 register ([Section 29.3.16](#)). This feature allows pins that are not used for DMM functionality to be used as GIO pins. Only the pins shown in [Table 29-5](#) can be used for a desired port width.

**Table 29-5. Pins Used for Data Communication**

Port Width	Pins Used
1	DMMDATA[0]
2	DMMDATA[1:0]
4	DMMDATA[3:0]
8	DMMDATA[7:0]
16	DMMDATA[15:0]

**NOTE:** If pins other than the ones specified in [Table 29-5](#) are programmed as functional pins for a desired port width, the received data will be corrupted and will not be transferred to the deserializer.

**NOTE:** If DMMCLK or DMMSYNC are programmed as nonfunctional pins, functional operation will not occur.

#### 29.2.2.1 Signal Description

**Table 29-6. Signal Description**

DMMSYNC	This signal has to be provided by external hardware. It signals the start of a new packet. It has to be active (high) for one full DMMCLK cycle, starting with the rising edge of DMMCLK. If the DMMSYNC pulse is longer than a single DMMCLK cycle and two falling edges of DMMCLK see a high pulse on DMMSYNC, the module will treat the second DMMSYNC pulse as the start of a packet and will flag a PACKET_ERR_INT (Section 29.3.5).
DMMCLK	The clock is externally generated and can be suspended between two packets. For this feature, CONTCLK must be set to 0 (Section 29.3.1). If the clock is not stopped between two packets, CONTCLK must be set to 1. Data will be latched on the falling edge of the DMMCLK signal.
DMMENA	This signal is pulled high if no new data should be received via the data pins, because of a potential overflow situation.
DMMDATA[15:0]	These pins receive the packet information transmitted by the external hardware. Data is latched on the falling edge of DMMCLK.

Figure 29-4 shows an example of multiple packets received during trace mode, in noncontinuous clock configuration.

**Figure 29-4. Packet Sync Signal Example**

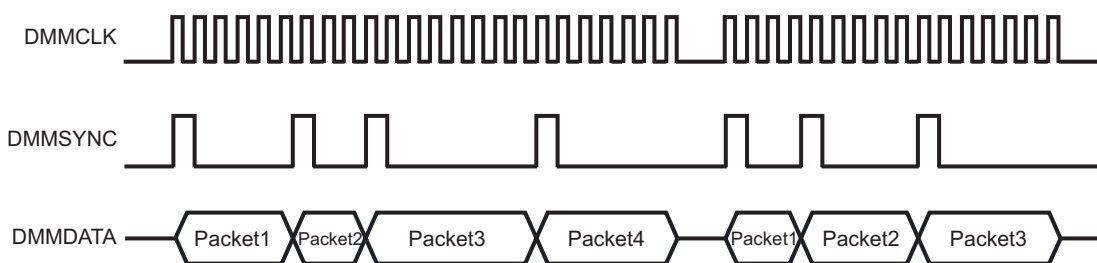
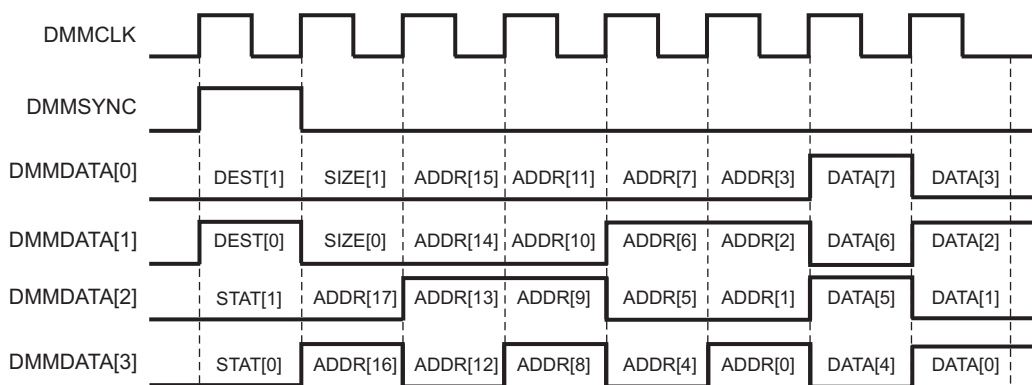


Figure 29-5 shows an example of a 4-bit data port with 8-bit receive data (A5h) to be written into DEST1 (address 0001 2345h) on a trace mode packet.

**Figure 29-5. Example Single Packet Transmission**



### 29.2.3 Error Handling

The module will generate two different kind of errors. Once an error condition is recognized, an interrupt will be generated if enabled.

#### 29.2.3.1 Overflow Error

This error is signaled when the module has received new data before the previous data was written to the destination address. If the internal buffers are full, the DMMENA signal will go high. If the sending module does not evaluate the DMMENA signal and keeps on sending new frames, the data that was previously received might be overwritten, thus resulting in setting the BUFF\_OVF flag (Section 29.3.5)

### 29.2.3.2 Packet Error

#### Noncontinuous Clock Mode

The size of the incoming packet is defined by the SIZE information of a trace mode packet or the programmed size of a direct data mode packet. If too many or less than the number of bits are received before the next sync signal, the PACKET\_ERR\_INT flag will be set (Section 29.3.5). In case of receiving a DMMCLK signal without a corresponding DMMSYNC signal, a packet error will also be generated.

#### Continuous Clock Mode

If less than the expected number of bits are received, the PACKET\_ERR\_INT flag will be set (Section 29.3.5) when the next DMMSYNC signal is received. Packets with more than the expected number of bits cannot be detected.

The check for packet error is done only after the detection of the first DMMSYNC signal after the DMM is turned on or comes out of suspend mode (with COS = 0; Section 29.3.1), i.e. before the reception of first DMMSYNC, the toggling of DMMCLK would be ignored.

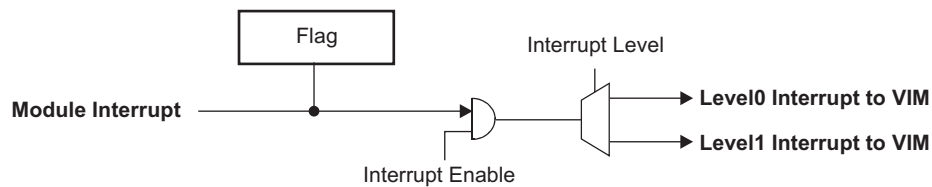
### 29.2.3.3 Bus Error

If an error occurs on the microcontroller internal bus system while transferring the data from the DMM to the destination, the BUSERROR flag will be set.

## 29.2.4 Interrupts

The module provides different interrupts. These can be programmed to different interrupt levels independently using DMMINTLVL (Section 29.3.4).

Figure 29-6. Interrupt Structure



Interrupts can be divided into error interrupts and functional interrupts. The error handling is described in Section 29.2.3. Functional interrupts depend on the mode (Trace Mode, Direct Data Mode) the DMM module is used in.

**Trace Mode:** An interrupt can be enabled whenever an access to the lowest address of a defined region is performed. This address is the starting address programmed in the DMMDESTxREGy register. An interrupt for each of the region can be generated by setting the individual interrupt enable bits.

**Direct Data Mode:** There are two interrupts which can be individually controlled. One is generated when the buffer pointer reaches the end of the defined buffer and wraps around (EO\_BUFF; Section 29.3.2). The other one is generated when the buffer pointer matches the programmed interrupt threshold (PROG\_BUFF; Section 29.3.2). The buffer pointer points to the next address to be written, therefore there are (interrupt threshold - 1) values stored in the buffer. The interrupt threshold can be programmed in the DMMINTPT register (Section 29.3.11).

### 29.3 Control Registers Summary

This section describes the DMM registers. The registers support 8, 16, and 32-bit writes. The offset is relative to the associated peripheral select. [Table 29-7](#) provides a summary of the registers and their bits. The base address of the DMM module is FFFF F700h.

**Table 29-7. DMM Registers Summary**

Offset	Acronym	Register Description	Section
0	DMMGLBCTRL	DMM Global Control Register	<a href="#">Section 29.3.1</a>
4h	DMMINTSET	DMM Interrupt Set Register	<a href="#">Section 29.3.2</a>
8h	DMMINTCLR	DMM Interrupt Clear Register	<a href="#">Section 29.3.3</a>
0Ch	DMMINTLVL	DMM Interrupt Level Register	<a href="#">Section 29.3.4</a>
10h	DMMINTFLG	DMM Interrupt Flag Register	<a href="#">Section 29.3.5</a>
14h	DMMOFF1	DMM Interrupt Offset 1 Register	<a href="#">Section 29.3.6</a>
16h	DMMOFF2	DMM Interrupt Offset 2 Register	<a href="#">Section 29.3.7</a>
1Ch	DMMDDMDEST	DMM Direct Data Mode Destination Register	<a href="#">Section 29.3.8</a>
20h	DMMDDMBL	DMM Direct Data Mode Blocksize Register	<a href="#">Section 29.3.9</a>
24h	DMMDDMPT	DMM Direct Data Mode Pointer Register	<a href="#">Section 29.3.10</a>
28h	DMMINTPT	DMM Direct Data Mode Interrupt Pointer Register	<a href="#">Section 29.3.11</a>
2Ch, 3Ch, 4Ch, 5Ch	DMMDESTxREG1	DMM Destination x Region 1	<a href="#">Section 29.3.12</a>
30h, 40h, 50h, 60h	DMMDESTxBL1	DMM Destination x Blocksize 1	<a href="#">Section 29.3.13</a>
34h, 44h, 54h, 64h	DMMDESTxREG2	DMM Destination x Region 2	<a href="#">Section 29.3.14</a>
38h, 48h, 58h, 68h	DMMDESTxBL2	DMM Destination x Blocksize 2	<a href="#">Section 29.3.15</a>
6Ch	DMMPC0	DMM Pin Control 0	<a href="#">Section 29.3.16</a>
70h	DMMPC1	DMM Pin Control 1	<a href="#">Section 29.3.17</a>
74h	DMMPC2	DMM Pin Control 2	<a href="#">Section 29.3.18</a>
78h	DMMPC3	DMM Pin Control 3	<a href="#">Section 29.3.19</a>
7Ch	DMMPC4	DMM Pin Control 4	<a href="#">Section 29.3.20</a>
80h	DMMPC5	DMM Pin Control 5	<a href="#">Section 29.3.21</a>
84h	DMMPC6	DMM Pin Control 6	<a href="#">Section 29.3.22</a>
88h	DMMPC7	DMM Pin Control 7	<a href="#">Section 29.3.23</a>
8Ch	DMMPC8	DMM Pin Control 8	<a href="#">Section 29.3.24</a>

### 29.3.1 DMM Global Control Register (DMMGLBCTRL)

With this register the basic operation of the module is selected.

**Figure 29-7. DMM Global Control Register (DMMGLBCTRL) [offset = 00h]**

31	Reserved			25	24
				BUSY	
R-0					
23	Reserved			18	17
				CONTCLK	COS
				RESET	16
R-0				R/W-0	R/W-0
15	Reserved			10	9
				DDM_WIDTH	
				TM_DDM	
R-0				R/W-0	R/W-0
7	Reserved			4	3
				ON/OFF	
R-0				R/W-5h	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 29-8. DMM Global Control Register (DMMGLBCTRL) Field Descriptions**

Bit	Field	Value	Description
31-25	Reserved	0	Reads returns 0. Writes have no effect.
24	BUSY	0	Busy indicator. The DMM does not currently receive data and has no data in its internal buffers, which needs to be transferred.
		1	The module is currently receiving data, or has data in its internal buffers
23-19	Reserved	0	Reads returns 0. Writes have no effect.
18	CONTCLK		Continuous DMMCLK input. User and privilege mode read, privilege mode write:
		0	DMMCLK is expected to be suspended between two packets.
		1	DMMCLK is expected to be free running between packets.
17	COS		Continue on suspend. Influences behavior of module while in debug mode. In all cases the corresponding interrupt will be set.
			User and privilege mode (read):
		0	Packets will not be received during debug mode. Before entering debug mode, the ongoing reception of a packet will be finished and the value will be written to the destination.
		1	Continue receiving packets and update destination, while in debug mode
			Privilege mode (write):
		0	Disable data reception while in debug mode
		1	Enable data reception while in debug mode
16	RESET		Reset. This bit resets the state machine and the registers to its reset value, except the RESET bit itself. It must be cleared by writing to it.
			User and privilege mode (read):
		0	No reset of DMM module
		1	Reset of DMM module
			Privilege mode (write):
		0	No reset of DMM module
		1	Reset DMM module to its reset state
15-11	Reserved	0	Reads returns 0. Writes have no effect.



**Table 29-8. DMM Global Control Register (DMMGLBCTRL) Field Descriptions (continued)**

Bit	Field	Value	Description
10-9	DDM_WIDTH	Bit Encoding 0 1h 2h 3h	Packet Width in direct data mode. User and privilege mode read and privilege mode write operation: Transfer Size 8 bit 16 bit 32 bit Reserved
8	TM_DMM	0 1 0 1	Packet Format. User and privilege mode (read): The DMM module assumes packets in trace mode definition The DMM module assumes packets in direct data mode definition Privilege mode (write): Enable trace mode Enable direct data mode
7-4	Reserved	0	Reads returns 0. Writes have no effect.
3-0	ON/OFF	All other Ah All other Ah	Switch module on or off User and privilege mode (read): The DMM module does not receive data The DMM module receives data and writes it to the buffer Privilege mode (write): Disable receive/write operations. Packets in reception, will still be finished Enable receive/write operations. Packets will be received 1 HCLK cycle after enabling the module

---

**NOTE:** It is recommended to write 5h to ON/OFF to avoid having a soft error inadvertently enabling the module when a single bit flips.

---



---

**NOTE:** Registers which affect the operation of the module, should be only programmed when the BUSY bit is 0 and the ON/OFF bits are not Ah.

---



---

**NOTE:** If the module was in operation, turned off (ON/OFF = all other than Ah) and then turned on (ON/OFF = Ah) again, it is recommended to perform a reset (RESET = 1) of the module before switching it on. This avoids that the state machine is held in an unrecoverable state.

---



---

**NOTE:** A write to these register bits while receiving a packet will not have any effect on the received packet. The mode change will be performed after the packet is received

---

### 29.3.2 DMM Interrupt Set Register (DMMINTSET)

This register contains the interrupt set bits for error interrupts and functional interrupts. Only the bits which are relevant for the particular mode (trace mode or direct data mode) will be taken into account for the interrupt generation.

**Figure 29-8. DMM Interrupt Set Register (DMMINTSET) [offset = 04h]**

31	30	29	28	27	26	25	24
Reserved							
R-0							
23	22	21	20	19	18	17	16
Reserved						PROG_BUFF	EO_BUFF
R-0						R/W-0	R/W-0
15	14	13	12	11	10	9	8
DEST3REG2	DEST3REG1	DEST2REG2	DEST2REG1	DEST1REG2	DEST1REG1	DEST0REG2	DEST0REG1
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
BUSERROR	BUFF_OVF	SRC_OVF	DEST3_ERR	DEST2_ERR	DEST1_ERR	DEST0_ERR	PACKET_ERR_INT
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 29-9. DMM Interrupt Set Register (DMMINTSET) Field Descriptions**

Bit	Field	Value	Description
31-18	Reserved	0	Reads returns 0. Writes have no effect.
17	PROG_BUFF	0 1	<b>Programmable Buffer Interrupt Set.</b> This enables the interrupt generation in case the buffer pointer equals the programmed value in the DMMINTPT register ( <a href="#">Section 29.3.11</a> ). This bit is only relevant in Direct Data Mode.  User and privilege mode (read): 0 No interrupt will be generated 1 An interrupt will be generated on pointer match.  Privilege mode (write): 0 No influence on bit 1 Enable interrupt (sets corresponding bit in DMMINTCLR; DMMINTLVL)
16	EO_BUFF	0 1	<b>End of Buffer Interrupt Set.</b> This enables the interrupt generation in case data was written to the last entry in the buffer and the pointer wrapped around to the beginning of the buffer. This bit is only relevant in Direct Data Mode.  User and privilege mode (read): 0 No interrupt will be generated 1 An interrupt will be generated on writing to the last entry.  Privilege mode (write): 0 No influence on bit 1 Enable interrupt (sets corresponding bit in DMMINTCLR; DMMINTLVL)
15	DEST3REG2	0 1	<b>Destination 3 Region 2 Interrupt Set.</b> This enables the interrupt generation in case data was accessed at the startaddress of Destination 3 Region 2. This bit is only relevant in Trace Mode.  User and privilege mode (read): 0 No interrupt will be generated 1 An interrupt will be generated on a write to the start address of this region  Privilege mode (write): 0 No influence on bit 1 Enable interrupt (sets corresponding bit in DMMINTCLR; DMMINTLVL)

**Table 29-9. DMM Interrupt Set Register (DMMINTSET) Field Descriptions (continued)**

Bit	Field	Value	Description
14	DEST3REG1	<p>0 1</p> <p>0 1</p>	<p><b>Destination 3 Region 1 Interrupt Set.</b> This enables the interrupt generation in case data was accessed at the startaddress of Destination 3 Region 1. This bit is only relevant in Trace Mode.</p> <p>User and privilege mode (read):</p> <p>No interrupt will be generated</p> <p>An interrupt will be generated on a write to the start address of this region</p> <p>Privilege mode (write):</p> <p>No influence on bit</p> <p>Enable interrupt (sets corresponding bit in DMMINTCLR; DMMINTLVL)</p>
13	DEST2REG2	<p>0 1</p> <p>0 1</p>	<p><b>Destination 2 Region 2 Interrupt Set.</b> This enables the interrupt generation in case data was accessed at the startaddress of Destination 2 Region 2. This bit is only relevant in Trace Mode.</p> <p>User and privilege mode (read):</p> <p>No interrupt will be generated</p> <p>An interrupt will be generated on a write to the start address of this region</p> <p>Privilege mode (write):</p> <p>No influence on bit</p> <p>Enable interrupt (sets corresponding bit in DMMINTCLR; DMMINTLVL)</p>
12	DEST2REG1	<p>0 1</p> <p>0 1</p>	<p><b>Destination 2 Region 1 Interrupt Set.</b> This enables the interrupt generation in case data was accessed at the startaddress of Destination 2 Region 1. This bit is only relevant in Trace Mode.</p> <p>User and privilege mode (read):</p> <p>No interrupt will be generated</p> <p>An interrupt will be generated on a write to the start address of this region</p> <p>Privilege mode (write):</p> <p>No influence on bit</p> <p>Enable interrupt (sets corresponding bit in DMMINTCLR; DMMINTLVL)</p>
11	DEST1REG2	<p>0 1</p> <p>0 1</p>	<p><b>Destination 1 Region 2 Interrupt Set.</b> This enables the interrupt generation in case data was accessed at the startaddress of Destination 1 Region 2. This bit is only relevant in Trace Mode.</p> <p>User and privilege mode (read):</p> <p>No interrupt will be generated</p> <p>An interrupt will be generated on a write to the start address of this region</p> <p>Privilege mode (write):</p> <p>No influence on bit</p> <p>Enable interrupt (sets corresponding bit in DMMINTCLR; DMMINTLVL)</p>
10	DEST1REG1	<p>0 1</p> <p>0 1</p>	<p><b>Destination 1 Region 1 Interrupt Set.</b> This enables the interrupt generation in case data was accessed at the startaddress of Destination 1 Region 1. This bit is only relevant in Trace Mode.</p> <p>User and privilege mode (read):</p> <p>No interrupt will be generated</p> <p>An interrupt will be generated on a write to the start address of this region</p> <p>Privilege mode (write):</p> <p>No influence on bit</p> <p>Enable interrupt (sets corresponding bit in DMMINTCLR; DMMINTLVL)</p>
9	DEST0REG2	<p>0 1</p> <p>0 1</p>	<p><b>Destination 0 Region 2 Interrupt Set.</b> This enables the interrupt generation in case data was accessed at the startaddress of Destination 0 Region 2. This bit is only relevant in Trace Mode.</p> <p>User and privilege mode (read):</p> <p>No interrupt will be generated</p> <p>An interrupt will be generated on a write to the start address of this region</p> <p>Privilege mode (write):</p> <p>No influence on bit</p> <p>Enable interrupt (sets corresponding bit in DMMINTCLR; DMMINTLVL)</p>

**Table 29-9. DMM Interrupt Set Register (DMMINTSET) Field Descriptions (continued)**

Bit	Field	Value	Description
8	DEST0REG1	<p>0 1</p> <p>0 1</p>	<p><b>Destination 0 Region 1 Interrupt Set.</b> This enables the interrupt generation in case data was accessed at the startaddress of Destination 0 Region 1. This bit is only relevant in Trace Mode.</p> <p>User and privilege mode (read):</p> <p>No interrupt will be generated</p> <p>An interrupt will be generated on a write to the start address of this region</p> <p>Privilege mode (write):</p> <p>No influence on bit</p> <p>Enable interrupt (sets corresponding bit in DMMINTCLR; DMMINTLVL)</p>
7	BUSERROR	<p>0 1</p> <p>0 1</p>	<p><b>Bus Error Response</b> for errors generated when doing internal bus transfers.</p> <p>User and privilege mode (read):</p> <p>No interrupt will be generated</p> <p>An interrupt will be generated</p> <p>Privilege mode (write):</p> <p>No influence on bit</p> <p>Enable interrupt (sets corresponding bit in DMMINTCLR; DMMINTLVL)</p>
6	BUFF_OVF	<p>0 1</p> <p>0 1</p>	<p><b>Buffer Overflow.</b> This enables the interrupt generation in case new data is received, while the previous data still has not been transmitted.</p> <p>User and privilege mode (read):</p> <p>No interrupt will be generated</p> <p>An interrupt will be generated</p> <p>Privilege mode (write):</p> <p>No influence on bit</p> <p>Enable interrupt (sets corresponding bit in DMMINTCLR; DMMINTLVL)</p>
5	SRC_OVF	<p>0 1</p> <p>0 1</p>	<p><b>Source Overflow.</b> This enables an interrupt if the external system experienced an overflow which was signalled in the Trace Mode packet.</p> <p>User and privilege mode (read):</p> <p>No interrupt will be generated</p> <p>An interrupt will be generated</p> <p>Privilege mode (write):</p> <p>No influence on bit</p> <p>Enable interrupt (sets corresponding bit in DMMINTCLR; DMMINTLVL)</p>
4	DEST3_ERR	<p>0 1</p> <p>0 1</p>	<p><b>Destination 3 Error.</b> This enables the interrupt generation in case data should be written into a address not specified by DMMDEST3REG1/DMMDEST3BL1 or DMMDEST3REG2/DMMDEST3BL2. If both block sizes are programmed to 0 or a reserved value, the interrupt will still be generated, the write to the internal RAM however will not take place.</p> <p>User and privilege mode (read):</p> <p>No interrupt will be generated</p> <p>An interrupt will be generated</p> <p>Privilege mode (write):</p> <p>No influence on bit</p> <p>Enable interrupt (sets corresponding bit in DMMINTCLR; DMMINTLVL)</p>

**Table 29-9. DMM Interrupt Set Register (DMMINTSET) Field Descriptions (continued)**

Bit	Field	Value	Description
3	DEST2_ERR		<p><b>Destination 2 Error Interrupt Set.</b> This enables the interrupt generation in case data should be written into a address not specified by DMMDEST2REG1/DMMDEST2BL1 or DMMDEST2REG2/DMMDEST2BL2. If both block sizes are programmed to 0 or a reserved value, the interrupt will still be generated, the write to the internal RAM however will not take place.</p> <p>User and privilege mode (read):</p> <p>0 No interrupt will be generated 1 An interrupt will be generated</p> <p>Privilege mode (write):</p> <p>0 No influence on bit 1 Enable interrupt (sets corresponding bit in DMMINTCLR; DMMINTLVL)</p>
2	DEST1_ERR		<p><b>Destination 1 Error Interrupt Set.</b> This enables the interrupt generation in case data should be written into a address not specified by DMMDEST1REG1/DMMDEST1BL1 or DMMDEST1REG2/DMMDEST1BL2. If both block sizes are programmed to 0 or a reserved value, the interrupt will still be generated, the write to the internal RAM however will not take place.</p> <p>User and privilege mode (read):</p> <p>0 No interrupt will be generated 1 An interrupt will be generated</p> <p>Privilege mode (write):</p> <p>0 No influence on bit 1 Enable interrupt (sets corresponding bit in DMMINTCLR; DMMINTLVL)</p>
1	DEST0_ERR		<p><b>Destination 0 Error Interrupt Set.</b> This enables the interrupt generation in case data should be written into a address not specified by DMMDEST0REG1/DMMDEST0BL1 or DMMDEST0REG2/DMMDEST0BL2. If both block sizes are programmed to 0 or a reserved value, the interrupt will still be generated, the write to the internal RAM however will not take place.</p> <p>User and privilege mode (read):</p> <p>0 No interrupt will be generated 1 An interrupt will be generated</p> <p>Privilege mode (write):</p> <p>0 No influence on bit 1 Enable interrupt (sets corresponding bit in DMMINTCLR; DMMINTLVL)</p>
0	PACKET_ERR_I NT		<p><b>Packet Error.</b> This enables the interrupt generation in case of an error condition in the packet reception. Please refer to <a href="#">Section 29.2.3</a> for the error conditions.</p> <p>User and privilege mode (read):</p> <p>0 No interrupt will be generated 1 An interrupt will be generated</p> <p>Privilege mode (write):</p> <p>0 No influence on bit 1 Enable interrupt (sets corresponding bit in DMMINTCLR; DMMINTLVL)</p>

### 29.3.3 DMM Interrupt Clear Register (DMMINTCLR)

This register contains the interrupt clear bits for error interrupts and functional interrupts. Only the bits which are relevant for the particular mode (trace mode or direct data mode) will be taken into account for the interrupt generation

**Figure 29-9. DMM Interrupt Clear Register (DMMINTCLR) [offset = 08h]**

31	30	29	28	27	26	25	24
Reserved							
R-0							
23	22	21	20	19	18	17	16
Reserved						PROG_BUFF	EO_BUFF
R-0						R/W-0	R/W-0
15	14	13	12	11	10	9	8
DEST3REG2	DEST3REG1	DEST2REG2	DEST2REG1	DEST1REG2	DEST1REG1	DEST0REG2	DEST0REG1
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
BUSERROR	BUFF_OVF	SRC_OVF	DEST3_ERR	DEST2_ERR	DEST1_ERR	DEST0_ERR	PACKET_ERR_INT
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 29-10. DMM Interrupt Clear Register (DMMINTCLR) Field Descriptions**

Bit	Field	Value	Description
31-18	Reserved	0	Reads returns 0. Writes have no effect.
17	PROG_BUFF	0 1	<p><b>Programmable Buffer Interrupt Set.</b> This disables the interrupt generation in case the buffer pointer equals the programmed value in the DMMINTPT register (<a href="#">Section 29.3.11</a>). This bit is only relevant in Direct Data Mode.</p> <p>User and privilege mode (read):</p> <p>0 No interrupt will be generated</p> <p>1 An interrupt will be generated on pointer match.</p> <p>Privilege mode (write):</p> <p>0 No influence on bit</p> <p>1 Disable interrupt (clears corresponding bit in DMMINTCLR; DMM Interrupt Level Register (DMMINTLVL))</p>
16	EO_BUFF	0 1	<p><b>End of Buffer Interrupt Set.</b> This disables the interrupt generation in case data was written to the last entry in the buffer and the pointer wrapped around to the beginning of the buffer. This bit is only relevant in Direct Data Mode.</p> <p>User and privilege mode (read):</p> <p>0 No interrupt will be generated</p> <p>1 An interrupt will be generated on writing to the last entry.</p> <p>Privilege mode (write):</p> <p>0 No influence on bit</p> <p>1 Disable interrupt (clears corresponding bit in DMMINTCLR; DMM Interrupt Level Register (DMMINTLVL))</p>

**Table 29-10. DMM Interrupt Clear Register (DMMINTCLR) Field Descriptions (continued)**

Bit	Field	Value	Description
15	DEST3REG2	<p>0</p> <p>1</p> <p>0</p> <p>1</p>	<p><b>Destination 3 Region 2 Interrupt Set.</b>This disables the interrupt generation in case data was accessed at the startaddress of Destination 3 Region 2. This bit is only relevant in Trace Mode.</p> <p>User and privilege mode (read):</p> <p>No interrupt will be generated</p> <p>An interrupt will be generated on a write to the start address of this region</p> <p>Privilege mode (write):</p> <p>No influence on bit</p> <p>Disable interrupt (clears corresponding bit in DMMINTCLR; DMM Interrupt Level Register (DMMINTLVL))</p>
14	DEST3REG1	<p>0</p> <p>1</p> <p>0</p> <p>1</p>	<p><b>Destination 3 Region 1 Interrupt Set.</b>This disables the interrupt generation in case data was accessed at the startaddress of Destination 3 Region 1. This bit is only relevant in Trace Mode.</p> <p>User and privilege mode (read):</p> <p>No interrupt will be generated</p> <p>An interrupt will be generated on a write to the start address of this region</p> <p>Privilege mode (write):</p> <p>No influence on bit</p> <p>Disable interrupt (clears corresponding bit in DMMINTCLR; DMM Interrupt Level Register (DMMINTLVL))</p>
13	DEST2REG2	<p>0</p> <p>1</p> <p>0</p> <p>1</p>	<p><b>Destination 2 Region 2 Interrupt Set.</b>This disables the interrupt generation in case data was accessed at the startaddress of Destination 2 Region 2. This bit is only relevant in Trace Mode.</p> <p>User and privilege mode (read):</p> <p>No interrupt will be generated</p> <p>An interrupt will be generated on a write to the start address of this region</p> <p>Privilege mode (write):</p> <p>No influence on bit</p> <p>Disable interrupt (clears corresponding bit in DMMINTCLR; DMM Interrupt Level Register (DMMINTLVL))</p>
12	DEST2REG1	<p>0</p> <p>1</p> <p>0</p> <p>1</p>	<p><b>Destination 2 Region 1 Interrupt Set.</b>This disables the interrupt generation in case data was accessed at the startaddress of Destination 2 Region 1. This bit is only relevant in Trace Mode.</p> <p>User and privilege mode (read):</p> <p>No interrupt will be generated</p> <p>An interrupt will be generated on a write to the start address of this region</p> <p>Privilege mode (write):</p> <p>No influence on bit</p> <p>Disable interrupt (clears corresponding bit in DMMINTCLR; DMM Interrupt Level Register (DMMINTLVL))</p>
11	DEST1REG2	<p>0</p> <p>1</p> <p>0</p> <p>1</p>	<p><b>Destination 1 Region 2 Interrupt Set.</b>This disables the interrupt generation in case data was accessed at the startaddress of Destination 1 Region 2. This bit is only relevant in Trace Mode.</p> <p>User and privilege mode (read):</p> <p>No interrupt will be generated</p> <p>An interrupt will be generated on a write to the start address of this region</p> <p>Privilege mode (write):</p> <p>No influence on bit</p> <p>Disable interrupt (clears corresponding bit in DMMINTCLR; DMM Interrupt Level Register (DMMINTLVL))</p>

**Table 29-10. DMM Interrupt Clear Register (DMMINTCLR) Field Descriptions (continued)**

Bit	Field	Value	Description
10	DEST1REG1	<p>0 1</p> <p>0 1</p>	<p><b>Destination 1 Region 1 Interrupt Set.</b>This enables the interrupt generation in case data was accessed at the startaddress of Destination 1 Region 1. This bit is only relevant in Trace Mode.</p> <p>User and privilege mode (read):</p> <p>No interrupt will be generated</p> <p>An interrupt will be generated on a write to the start address of this region</p> <p>Privilege mode (write):</p> <p>No influence on bit</p> <p>Disable interrupt (clears corresponding bit in DMMINTCLR; DMM Interrupt Level Register (DMMINTLVL))</p>
9	DEST0REG2	<p>0 1</p> <p>0 1</p>	<p><b>Destination 0 Region 2 Interrupt Set.</b>This disables the interrupt generation in case data was accessed at the startaddress of Destination 0 Region 2. This bit is only relevant in Trace Mode.</p> <p>User and privilege mode (read):</p> <p>No interrupt will be generated</p> <p>An interrupt will be generated on a write to the start address of this region</p> <p>Privilege mode (write):</p> <p>No influence on bit</p> <p>Disable interrupt (clears corresponding bit in DMMINTCLR; DMM Interrupt Level Register (DMMINTLVL))</p>
8	DEST0REG1	<p>0 1</p> <p>0 1</p>	<p><b>Destination 0 Region 1 Interrupt Set.</b>This disables the interrupt generation in case data was accessed at the startaddress of Destination 0 Region 1. This bit is only relevant in Trace Mode.</p> <p>User and privilege mode (read):</p> <p>No interrupt will be generated</p> <p>An interrupt will be generated on a write to the start address of this region</p> <p>Privilege mode (write):</p> <p>No influence on bit</p> <p>Disable interrupt (clears corresponding bit in DMMINTCLR; DMM Interrupt Level Register (DMMINTLVL))</p>
7	BUSERROR	<p>0 1</p> <p>0 1</p>	<p><b>Bus Error Response</b> for errors generated when doing internal bus transfers.</p> <p>User and privilege mode (read):</p> <p>No interrupt will be generated</p> <p>An interrupt will be generated</p> <p>Privilege mode (write):</p> <p>No influence on bit</p> <p>Disable interrupt (clears corresponding bit in DMMINTCLR; DMM Interrupt Level Register (DMMINTLVL))</p>
6	BUFF_OVF	<p>0 1</p> <p>0 1</p>	<p><b>Buffer Overflow.</b>This disables the interrupt generation in case new data is received, while the previous data still has not been transmitted.</p> <p>User and privilege mode (read):</p> <p>No interrupt will be generated</p> <p>An interrupt will be generated</p> <p>Privilege mode (write):</p> <p>No influence on bit</p> <p>Disable interrupt (clears corresponding bit in DMMINTCLR; DMM Interrupt Level Register (DMMINTLVL))</p>



**Table 29-10. DMM Interrupt Clear Register (DMMINTCLR) Field Descriptions (continued)**

Bit	Field	Value	Description
5	SRC_OVF		<p><b>Source Overflow.</b> This disables an interrupt if the external system experienced an overflow which was signalled in the Trace Mode packet.</p> <p>User and privilege mode (read):</p> <p>0 No interrupt will be generated</p> <p>1 An interrupt will be generated</p> <p>Privilege mode (write):</p> <p>0 No influence on bit</p> <p>1 Disable interrupt (clears corresponding bit in DMMINTCLR; DMM Interrupt Level Register (DMMINTLVL))</p>
4	DEST3_ERR		<p><b>Destination 3 Error.</b> This disables the interrupt generation in case data should be written into an address not specified by DMMDEST3REG1/DMMDEST3BL1 or DMMDEST3REG2/DMMDEST3BL2. If both block sizes are programmed to 0 or a reserved value, the interrupt will still be generated, the write to the internal RAM however will not take place.</p> <p>User and privilege mode (read):</p> <p>0 No interrupt will be generated</p> <p>1 An interrupt will be generated</p> <p>Privilege mode (write):</p> <p>0 No influence on bit</p> <p>1 Disable interrupt (clears corresponding bit in DMMINTCLR; DMM Interrupt Level Register (DMMINTLVL))</p>
3	DEST2_ERR		<p><b>Destination 2 Error Interrupt Set.</b> This disables the interrupt generation in case data should be written into an address not specified by DMMDEST2REG1/DMMDEST2BL1 or DMMDEST2REG2/DMMDEST2BL2. If both block sizes are programmed to 0 or a reserved value, the interrupt will still be generated, the write to the internal RAM however will not take place.</p> <p>User and privilege mode (read):</p> <p>0 No interrupt will be generated</p> <p>1 An interrupt will be generated</p> <p>Privilege mode (write):</p> <p>0 No influence on bit</p> <p>1 Disable interrupt (clears corresponding bit in DMMINTCLR; DMM Interrupt Level Register (DMMINTLVL))</p>
2	DEST1_ERR		<p><b>Destination 1 Error Interrupt Set.</b> This disables the interrupt generation in case data should be written into an address not specified by DMMDEST1REG1/DMMDEST1BL1 or DMMDEST1REG2/DMMDEST1BL2. If both block sizes are programmed to 0 or a reserved value, the interrupt will still be generated, the write to the internal RAM however will not take place.</p> <p>User and privilege mode (read):</p> <p>0 No interrupt will be generated</p> <p>1 An interrupt will be generated</p> <p>Privilege mode (write):</p> <p>0 No influence on bit</p> <p>1 Disable interrupt (clears corresponding bit in DMMINTCLR; DMM Interrupt Level Register (DMMINTLVL))</p>

**Table 29-10. DMM Interrupt Clear Register (DMMINTCLR) Field Descriptions (continued)**

Bit	Field	Value	Description
1	DEST0_ERR		<p><b>Destination 0 Error Interrupt Set.</b>This disables the interrupt generation in case data should be written into a address not specified by DMMDEST0REG1/DMMDEST0BL1 or DMMDEST0REG2/DMMDEST0BL2. If both blocksizes are programmed to 0 or a reserved value, the interrupt will still be generated, the write to the internal RAM however will not take place.</p> <p>User and privilege mode (read):</p> <p>0 No interrupt will be generated 1 An interrupt will be generated</p> <p>Privilege mode (write):</p> <p>0 No influence on bit 1 Disable interrupt (clears corresponding bit in DMMINTCLR; DMM Interrupt Level Register (DMMINTLVL))</p>
0	PACKET_ERR_I NT		<p><b>Packet Error.</b>This disables the interrupt generation in case of an error condition in the packet reception. Please refer to <a href="#">Section 29.2.3</a> for the error conditions.</p> <p>User and privilege mode (read):</p> <p>0 No interrupt will be generated 1 An interrupt will be generated</p> <p>Privilege mode (write):</p> <p>0 No influence on bit 1 Disable interrupt (clears corresponding bit in DMMINTCLR; DMM Interrupt Level Register (DMMINTLVL))</p>

### 29.3.4 DMM Interrupt Level Register (DMMINTLVL)

This register contains the interrupt level bits for error interrupts and normal interrupts.

**Figure 29-10. DMM Interrupt Level Register (DMMINTLVL) [offset = 0Ch]**

31	30	29	28	27	26	25	24
Reserved							
R-0							
23	22	21	20	19	18	17	16
Reserved						PROG_BUFF	EO_BUFF
R-0						R/W-0	R/W-0
15	14	13	12	11	10	9	8
DEST3REG2	DEST3REG1	DEST2REG2	DEST2REG1	DEST1REG2	DEST1REG1	DEST0REG2	DEST0REG1
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
BUSERROR	BUFF_OVF	SRC_OVF	DEST3_ERR	DEST2_ERR	DEST1_ERR	DEST0_ERR	PACKET_ERR_INT
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 29-11. DMM Interrupt Level Register (DMMINTLVL) Field Descriptions**

Bit	Field	Value	Description
31-18	Reserved	0	Reads returns 0. Writes have no effect.
17	PROG_BUFF	0 1	<b>Programmable Buffer Interrupt Level</b> User and privilege mode read, privilege mode write: Interrupt mapped to level 0 Interrupt mapped to level 1
16	EO_BUFF	0 1	<b>End of Buffer Interrupt Level</b> User and privilege mode read, privilege mode write: Interrupt mapped to level 0 Interrupt mapped to level 1
15	DEST3REG2	0 1	<b>Destination 3 Region 2 Interrupt Level</b> User and privilege mode read, privilege mode write: Interrupt mapped to level 0 Interrupt mapped to level 1
14	DEST3REG1	0 1	<b>Destination 3 Region 1 Interrupt Level</b> User and privilege mode read, privilege mode write: Interrupt mapped to level 0 Interrupt mapped to level 1
13	DEST2REG2	0 1	<b>Destination 2 Region 2 Interrupt Level</b> User and privilege mode read, privilege mode write: Interrupt mapped to level 0 Interrupt mapped to level 1
12	DEST2REG1	0 1	<b>Destination 2 Region 1 Interrupt Level</b> User and privilege mode read, privilege mode write: Interrupt mapped to level 0 Interrupt mapped to level 1
11	DEST1REG2	0 1	<b>Destination 1 Region 2 Interrupt Level</b> User and privilege mode read, privilege mode write: Interrupt mapped to level 0 Interrupt mapped to level 1

**Table 29-11. DMM Interrupt Level Register (DMMINTLVL) Field Descriptions (continued)**

Bit	Field	Value	Description
10	DEST1REG1	0 1	<b>Destination 1 Region 1 Interrupt Level</b> User and privilege mode read, privilege mode write: Interrupt mapped to level 0 Interrupt mapped to level 1
9	DEST0REG2	0 1	<b>Destination 0 Region 2 Interrupt Level</b> User and privilege mode read, privilege mode write: Interrupt mapped to level 0 Interrupt mapped to level 1
8	DEST0REG1	0 1	<b>Destination 0 Region 1 Interrupt Level</b> User and privilege mode read, privilege mode write: Interrupt mapped to level 0 Interrupt mapped to level 1
7	BUSERROR	0 1	<b>BMM Bus Error Response</b> User and privilege mode read, privilege mode write: Interrupt mapped to level 0 Interrupt mapped to level 1
6	BUFF_OVF	0 1	<b>Write Buffer Overflow Interrupt Level</b> User and privilege mode read, privilege mode write: Interrupt mapped to level 0 Interrupt mapped to level 1
5	SRC_OVF	0 1	<b>Source Overflow Interrupt Level</b> User and privilege mode read, privilege mode write: Interrupt mapped to level 0 Interrupt mapped to level 1
4	DEST3_ERR	0 1	<b>Destination 3 Error Interrupt Level</b> User and privilege mode read, privilege mode write: Interrupt mapped to level 0 Interrupt mapped to level 1
3	DEST2_ERR	0 1	<b>Destination 2 Error Interrupt Level</b> User and privilege mode read, privilege mode write: Interrupt mapped to level 0 Interrupt mapped to level 1
2	DEST1_ERR	0 1	<b>Destination 1 Error Interrupt Level</b> User and privilege mode read, privilege mode write: Interrupt mapped to level 0 Interrupt mapped to level 1
1	DEST0_ERR	0 1	<b>Destination 0 Error Interrupt Level</b> User and privilege mode read, privilege mode write: Interrupt mapped to level 0 Interrupt mapped to level 1
0	PACKET_ERR_I NT	0 1	<b>Packet Error Interrupt Level</b> User and privilege mode read, privilege mode write: Interrupt mapped to level 0 Interrupt mapped to level 1

### 29.3.5 DMM Interrupt Flag Register (DMMINTFLG)

This register contains the interrupt level bits for error interrupts and normal interrupts.

**Figure 29-11. DMM Interrupt Flag Register (DMMINTFLG) [offset = 10h]**

31	30	29	28	27	26	25	24
Reserved							
R-0							
23	22	21	20	19	18	17	16
Reserved						PROG_BUFF	EO_BUFF
R-0						R/W-0	R/W-0
15	14	13	12	11	10	9	8
DEST3REG2	DEST3REG1	DEST2REG2	DEST2REG1	DEST1REG2	DEST1REG1	DEST0REG2	DEST0REG1
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
BUSERROR	BUFF_OVF	SRC_OVF	DEST3_ERR	DEST2_ERR	DEST1_ERR	DEST0_ERR	PACKET_ERR_INT
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 29-12. DMM Interrupt Flag Register (DMMINTFLG) Field Descriptions**

Bit	Field	Value	Description
31-18	Reserved	0	Reads returns 0. Writes have no effect.
17	PROG_BUFF		<b>Programmable Buffer Interrupt Flag</b> User and privilege mode (read): 0 No interrupt occurred 1 Interrupt occurred Privilege mode (write): 0 No influence on bit 1 Bit will be cleared
16	EO_BUFF		<b>End of Buffer Interrupt Flag</b> User and privilege mode (read): 0 No interrupt occurred 1 Interrupt occurred Privilege mode (write): 0 No influence on bit 1 Bit will be cleared
15	DEST3REG2		<b>Destination 3 Region 2 Interrupt Flag</b> User and privilege mode (read): 0 No interrupt occurred 1 Interrupt occurred Privilege mode (write): 0 No influence on bit 1 Bit will be cleared
14	DEST3REG1		<b>Destination 3 Region 1 Interrupt Flag</b> User and privilege mode (read): 0 No interrupt occurred 1 Interrupt occurred Privilege mode (write): 0 No influence on bit 1 Bit will be cleared

**Table 29-12. DMM Interrupt Flag Register (DMMINTFLG) Field Descriptions (continued)**

Bit	Field	Value	Description
13	DEST2REG2		<b>Destination 2 Region 2 Interrupt Flag</b> User and privilege mode (read): 0 No interrupt occurred 1 Interrupt occurred Privilege mode (write): 0 No influence on bit 1 Bit will be cleared
12	DEST2REG1		<b>Destination 2 Region 1 Interrupt Flag</b> User and privilege mode (read): 0 No interrupt occurred 1 Interrupt occurred Privilege mode (write): 0 No influence on bit 1 Bit will be cleared
11	DEST1REG2		<b>Destination 1 Region 2 Interrupt Flag</b> User and privilege mode (read): 0 No interrupt occurred 1 Interrupt occurred Privilege mode (write): 0 No influence on bit 1 Bit will be cleared
10	DEST1REG1		<b>Destination 1 Region 1 Interrupt Flag</b> User and privilege mode (read): 0 No interrupt occurred 1 Interrupt occurred Privilege mode (write): 0 No influence on bit 1 Bit will be cleared
9	DEST0REG2		<b>Destination 0 Region 2 Interrupt Flag</b> User and privilege mode (read): 0 No interrupt occurred 1 Interrupt occurred Privilege mode (write): 0 No influence on bit 1 Bit will be cleared
8	DEST0REG1		<b>Destination 0 Region 1 Interrupt Flag</b> User and privilege mode (read): 0 No interrupt occurred 1 Interrupt occurred Privilege mode (write): 0 No influence on bit 1 Bit will be cleared

**Table 29-12. DMM Interrupt Flag Register (DMMINTFLG) Field Descriptions (continued)**

Bit	Field	Value	Description
7	BUSERROR	0 1 0 1	<b>BMM Bus Error Response.</b> User and privilege mode (read): No interrupt occurred Interrupt occurred Privilege mode (write): No influence on bit Bit will be cleared
6	BUFF_OVF	0 1 0 1	<b>Write Buffer Overflow Interrupt Flag</b> User and privilege mode (read): No interrupt occurred Interrupt occurred Privilege mode (write): No influence on bit Bit will be cleared
5	SRC_OVF	0 1 0 1	<b>Source Overflow Interrupt Flag</b> User and privilege mode (read): No interrupt occurred Interrupt occurred Privilege mode (write): No influence on bit Bit will be cleared
4	DEST3_ERR	0 1 0 1	<b>Destination 3 Error Interrupt Flag</b> User and privilege mode (read): No interrupt occurred Interrupt occurred Privilege mode (write): No influence on bit Bit will be cleared
3	DEST2_ERR	0 1 0 1	<b>Destination 2 Error Interrupt Flag</b> User and privilege mode (read): No interrupt occurred Interrupt occurred Privilege mode (write): No influence on bit Bit will be cleared
2	DEST1_ERR	0 1 0 1	<b>Destination 1 Error Interrupt Flag</b> User and privilege mode (read): No interrupt occurred Interrupt occurred Privilege mode (write): No influence on bit Bit will be cleared

**Table 29-12. DMM Interrupt Flag Register (DMMINTFLG) Field Descriptions (continued)**

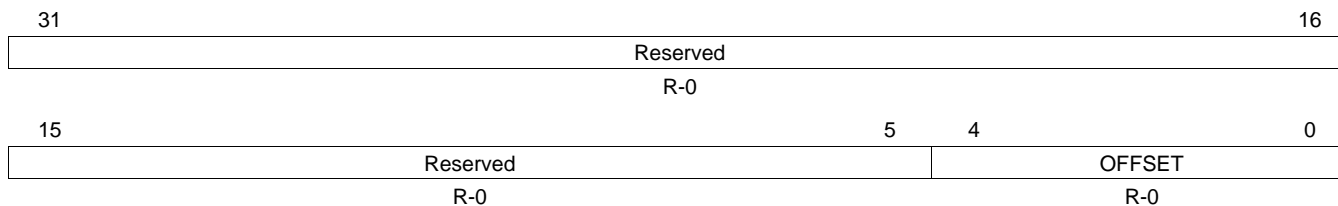
Bit	Field	Value	Description
1	DEST0_ERR		<b>Destination 0 Error Interrupt Flag</b>
			User and privilege mode (read):
		0	No interrupt occurred
		1	Interrupt occurred
			Privilege mode (write):
		0	No influence on bit
		1	Bit will be cleared
0	PACKET_ERR_I NT		<b>Packet Error Interrupt Flag</b>
			User and privilege mode (read):
		0	No interrupt occurred
		1	Interrupt occurred
			Privilege mode (write):
		0	No influence on bit
		1	Bit will be cleared



### 29.3.6 DMM Interrupt Offset 1 Register (DMMOFF1)

This register holds the offset indicating which interrupt occurred on interrupt level 0. The CPU can read this register to determine the source of the interrupt without having to test individual interrupt flags.

**Figure 29-12. DMM Interrupt Offset 1 Register (DMMOFF1) [offset = 14h]**



LEGEND: R = Read only; -n = value after reset

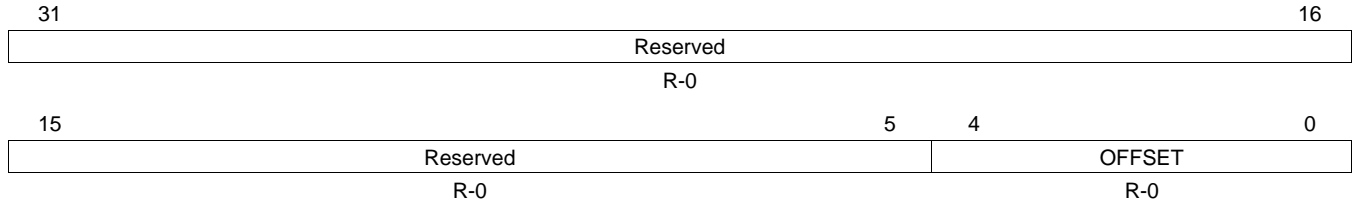
**Table 29-13. DMM Interrupt Offset 1 Register (DMMOFF1) Field Descriptions**

Bit	Field	Value	Description
31-5	Reserved	0	Read returns 0. Writes have no effect.
4-0	OFFSET	Bit Encoding	User and privilege mode (read): Interrupt
		0	Phantom. All interrupt flags have been cleared before the offset register has been read.
		1h	Packet Error
		2h	Destination 0 Error
		3h	Destination 1 Error
		4h	Destination 2 Error
		5h	Destination 3 Error
		6h	Source Overflow
		7h	Buffer Overflow
		8h	Bus Error
		9h	Destination 0 Region 1
		Ah	Destination 0 Region 2
		Bh	Destination 1 Region 1
		Ch	Destination 1 Region 2
		Dh	Destination 2 Region 1
		Eh	Destination 2 Region 2
		Fh	Destination 3 Region 1
		10h	Destination 3 Region 2
		11h	End of Buffer
		12h	Programmable Buffer
		13h - 1Fh	Reserved
			Reading the offset will clear the corresponding flag in DMMINTFLG ( <a href="#">Section 29.3.5</a> ).
			Privilege and user mode writes have no effect

**29.3.7 DMM Interrupt Offset 2 Register (DMMOFF2)**

This register holds the offset indicating which interrupt occurred on interrupt level 1. The CPU can read this register to determine the source of the interrupt without having to test individual interrupt flags.

**Figure 29-13. DMM Interrupt Offset 2 Register (DMMOFF2) [offset = 18h]**



LEGEND: R = Read only; -n = value after reset

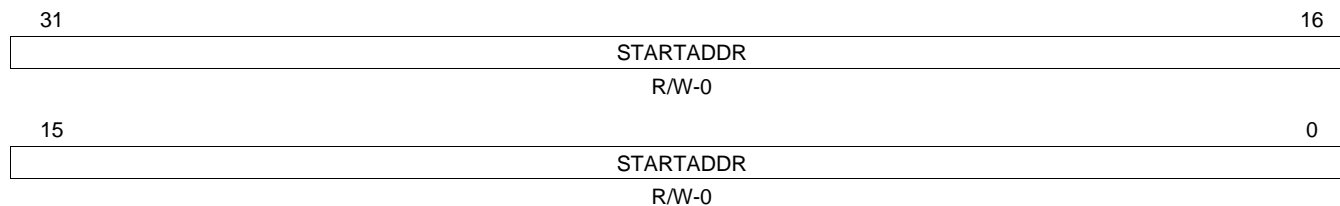
**Table 29-14. DMM Interrupt Offset 2 Register (DMMOFF1) Field Descriptions**

Bit	Field	Value	Description
31-5	Reserved	0	Read returns 0. Writes have no effect.
4-0	OFFSET	Bit Encoding 0 1h 2h 3h 4h 5h 6h 7h 8h 9h Ah Bh Ch Dh Eh Fh 10h 11h 12h 13h - 1Fh	User and privilege mode (read): Interrupt Phantom. All interrupt flags have been cleared before the offset register has been read. Packet Error Destination 0 Error Destination 1 Error Destination 2 Error Destination 3 Error Source Overflow Buffer Overflow Bus Error Destination 0 Region 1 Destination 0 Region 2 Destination 1 Region 1 Destination 1 Region 2 Destination 2 Region 1 Destination 2 Region 2 Destination 3 Region 1 Destination 3 Region 2 End of Buffer Programmable Buffer Reserved Reading the offset will clear the corresponding flag in DMMINTFLG ( <a href="#">Section 29.3.5</a> ). Privilege and user mode writes have no effect

### 29.3.8 DMM Direct Data Mode Destination Register (DMMDDMDEST)

This register defines the starting address of the buffer used to store the received data in Direct Data Mode. By writing to this register, the DMMDDMPT register ([Section 29.3.10](#)) will be set to 0x0000.

**Figure 29-14. DMM Direct Data Mode Destination Register (DMMDDMDEST) [offset = 1Ch]**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

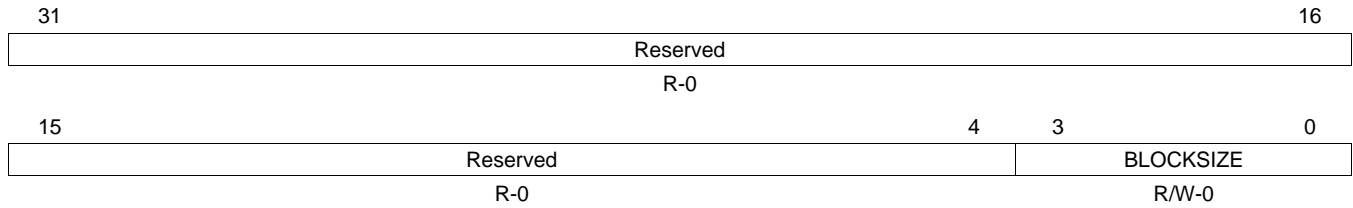
**Table 29-15. DMM Direct Data Mode Destination Register (DMMDDMDEST) Field Descriptions**

Bit	Field	Value	Description
31-0	STARTADDR		<p>These bits define the starting address of the buffer. The starting address has to be a multiple of the blocksize chosen in DMMDDMBL (<a href="#">Section 29.3.9</a>).</p> <p>User and privilege mode (read): current start address</p> <p>Privilege mode (write): sets start address to value written</p>

**29.3.9 DMM Direct Data Mode Blocksize Register (DMMDDMBL)**

This register defines the blocksize of the buffer used to store the received data in Direct Data Mode.

**Figure 29-15. DMM Direct Data Mode Blocksize Register (DMMDDMBL) [offset = 20h]**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

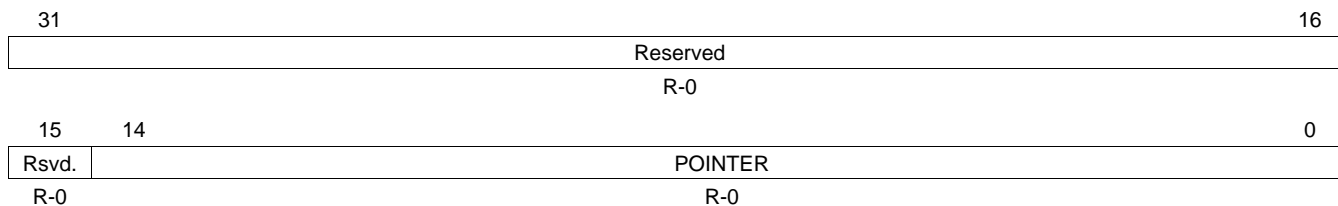
**Table 29-16. DMM Direct Data Mode Blocksize Register (DMMDDMBL) Field Descriptions**

Bit	Field	Value	Description
31-4	Reserved	0	Read returns 0. Writes have no effect.
3-0	BLOCKSIZE		These bits define the size of the buffer region User and privilege mode (read): current block size Privilege mode (write): Buffer disabled. No data will be stored. 32 Byte 64 Byte 128 Byte 256 Byte 512 Byte 1 KByte 2 KByte 4 KByte 8 KByte 16 KByte 32 KByte Reserved

### 29.3.10 DMM Direct Data Mode Pointer Register (DMMDDMPT)

This register shows the pointer into the buffer programmed by DMMDDMDEST (Section 29.3.8) and DMMDDMBL (Section 29.3.9).

**Figure 29-16. DMM Direct Data Mode Pointer Register (DMMDDMPT) [offset = 24h]**



LEGEND: R = Read only; -n = value after reset

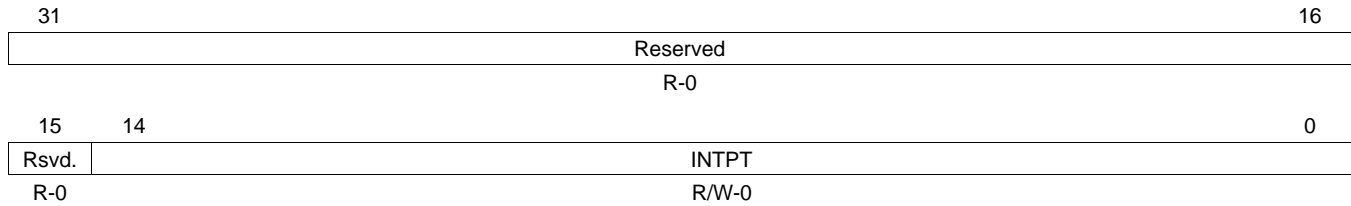
**Table 29-17. DMM Direct Data Mode Pointer Register (DMMDDMPT) Field Descriptions**

Bit	Field	Value	Description
31-15	Reserved	0	Read returns 0. Writes have no effect.
14-0	POINTER		These bits hold the pointer to the next entry to be written in the buffer. The pointer points to the byte aligned address. If in 16-bit DDM mode, bit 0 will be 0. If in 32-bit DDM mode, bit 0 and 1 will be 0.  User and privilege mode (read): next data entry  Privilege mode (write): writes have no effect

### 29.3.11 DMM Direct Data Mode Interrupt Pointer Register (DMMINTPT)

This register can be programmed to hold a threshold to which the DMMDDMPT register ([Section 29.3.10](#)) is compared. An interrupt can be generated when both match.

**Figure 29-17. DMM Direct Data Mode Interrupt Pointer Register (DMMINTPT) [offset = 28h]**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

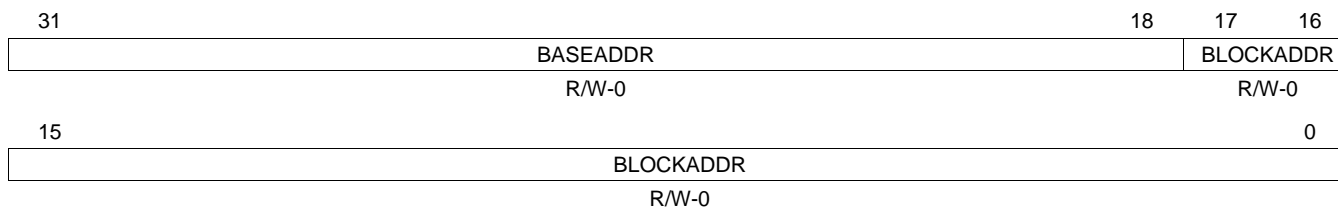
**Table 29-18. DMM Direct Data Mode Interrupt Pointer Register (DMMINTPT) Field Descriptions**

Bit	Field	Value	Description
31-15	Reserved	0	Read returns 0. Writes have no effect.
14-0	INTPT		Interrupt Pointer. When the buffer pointer ( <a href="#">Section 29.3.10</a> ) matches the programmed value in DMMINTPT and the PROG_BUF interrupt ( <a href="#">Section 29.3.2</a> ) is set, an interrupt is generated.  User and privilege mode (read): current interrupt threshold  Privilege mode (write): new interrupt threshold

### 29.3.12 DMM Destination x Region 1 (DMMDESTxREG1)

This register defines the starting address of the buffer used to store the received data in Trace Mode. If the received data does not fall into the address range defined by DMMDESTxREG1 and DMMDESTxBL1, an interrupt (DESTx\_ERR) can be generated. The description below is valid for following registers: DMMDEST0REG1, DMMDEST1REG1, DMMDEST2REG1, DMMDEST3REG1.

**Figure 29-18. DMM Destination x Region 1 (DMMDESTxREG1) [offset = 2Ch, 3Ch, 4Ch, 5Ch]**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

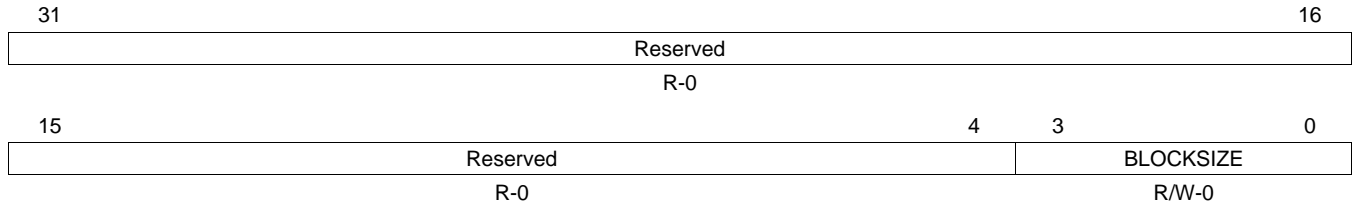
**Table 29-19. DMM Destination x Region 1 (DMMDESTxREG1) Field Descriptions**

Bit	Field	Value	Description
31-18	BASEADDR		These bits define the base address of the 256kB region where the buffer is located. User and privilege mode (read): current start address Privilege mode (write): sets base address to value written
17-0	BLOCKADDR		These bits define the starting address of the buffer in the 256kB page. The starting address has to be a multiple of the blocksize chosen in DMMDESTxBL1 ( <a href="#">Section 29.3.13</a> ). User and privilege mode (read): current start address Privilege mode (write): sets start address to value written

**29.3.13 DMM Destination x Blocksize 1 (DMMDESTxBL1)**

This register defines the blocksize of the buffer used to store the received data in Trace Mode. If the received data does not fall into the address range defined by DMMDESTxREG1 and DMMDESTxBL1, an interrupt (DESTx\_ERR) can be generated. The description below is valid for following registers: DMMDEST0BL1, DMMDEST1BL1, DMMDEST2BL1, DMMDEST3BL1.

**Figure 29-19. DMM Destination x Blocksize 1 (DMMDESTxBL1) [offset = 30h, 40h, 50h, 60h]**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 29-20. DMM Destination x Blocksize 1 (DMMDESTxBL1) Field Descriptions**

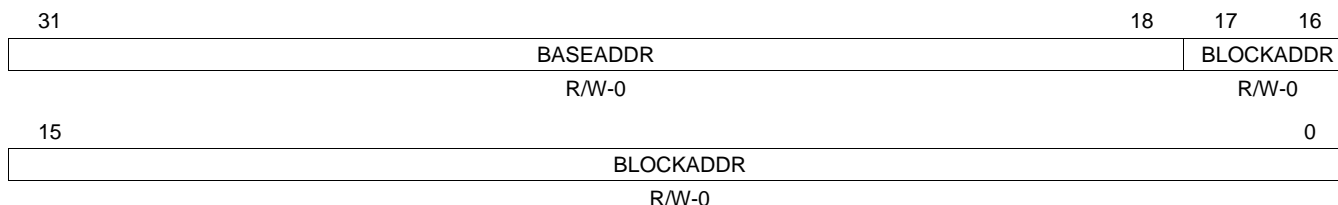
Bit	Field	Value	Description
31-4	Reserved	0	Read returns 0. Writes have no effect.
3-0	BLOCKSIZE		These bits define the length of the buffer region. If all bits are 0, the region is disabled and no data will be stored. User and privilege mode (read): current block size Privilege mode (write):
		0	Region disabled
		1h	1 KByte
		2h	2 KByte
		3h	4 KByte
		4h	8 KByte
		5h	16 KByte
		6h	32 KByte
		7h	64 KByte
		8h	128 KByte
		9h	256 KByte
		Ah - Fh	Reserved



### 29.3.14 DMM Destination x Region 2 (DMMDESTxREG2)

This register defines the starting address of the buffer used to store the received data in Trace Mode. If the received data does not fall into the address range defined by DMMDESTxREG2 and DMMDESTxBL2, an interrupt (DESTx\_ERR) can be generated. The description below is valid for following registers: DMMDEST0REG2, DMMDEST1REG2, DMMDEST2REG2, DMMDEST3REG2.

**Figure 29-20. DMM Destination x Region 2 (DMMDESTxREG2) [offset = 34h, 44h, 54h, 64h]**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

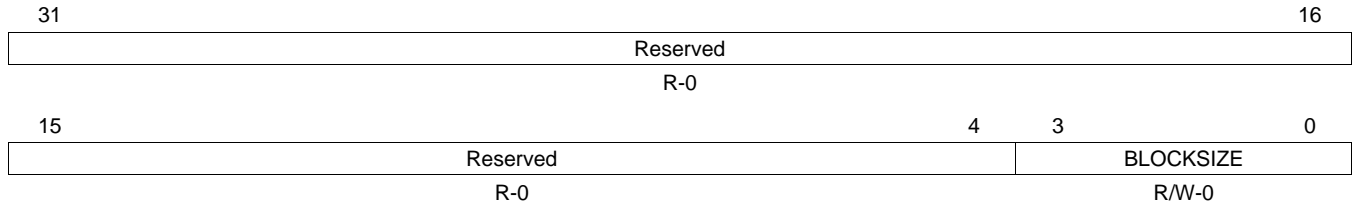
**Table 29-21. DMM Destination x Region 2 (DMMDESTxREG2) Field Descriptions**

Bit	Field	Value	Description
31-18	BASEADDR		These bits define the base address of the 256kB region where the buffer is located. User and privilege mode (read): current start address Privilege mode (write): sets base address to value written
17-0	BLOCKADDR		These bits define the starting address of the buffer in the 256kB page. The starting address has to be a multiple of the blocksize chosen in DMMDESTxBL1 ( <a href="#">Section 29.3.15</a> ). User and privilege mode (read): current start address Privilege mode (write): sets start address to value written

**29.3.15 DMM Destination x Blocksize 2 (DMMDESTxBL2)**

This register defines the blocksize of the buffer used to store the received data in Trace Mode. If the received data does not fall into the address range defined by DMMDESTxREG2 and DMMDESTxBL2, an interrupt (DESTx\_ERR) can be generated. The description below is valid for following registers: DMMDEST0BL2, DMMDEST1BL2, DMMDEST2BL2, DMMDEST3BL2.

**Figure 29-21. DMM Destination x Blocksize 2 (DMMDESTxBL2) [offset = 38h, 48h, 58h, 68h]**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 29-22. DMM Destination x Blocksize 2 (DMMDESTxBL2) Field Descriptions**

Bit	Field	Value	Description
31-4	Reserved	0	Read returns 0. Writes have no effect.
3-0	BLOCKSIZE		These bits define the length of the buffer region. If all bits are 0, the region is disabled and no data will be stored. User and privilege mode (read): current block size Privilege mode (write):
		0	Region disabled
		1h	1 KByte
		2h	2 KByte
		3h	4 KByte
		4h	8 KByte
		5h	16 KByte
		6h	32 KByte
		7h	64 KByte
		8h	128 KByte
		9h	256 KByte
		Ah - Fh	Reserved

### 29.3.16 DMM Pin Control 0 (DMMPC0)

This register defines if the DMM pins are used in functional or GIO mode. It should only be written when ON/OFF = 0101 and the BUSY bit = 0 (Section 29.3.1). If pins other than the pins specified in Table 29-5 are configured, or DMMCLK and DMMSYNC are programmed as non-functional pins, no operation in trace mode or direct data mode is possible.

**Figure 29-22. DMM Pin Control 0 (DMMPC0) [offset = 6Ch]**

31	30	29	28	27	26	25	24
Reserved							
R-0							
23	22	21	20	19	18	17	16
Reserved					ENAFUNC	DATA15FUNC	DATA14FUNC
R-0					R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
DATA13FUNC	DATA12FUNC	DATA11FUNC	DATA10FUNC	DATA9FUNC	DATA8FUNC	DATA7FUNC	DATA6FUNC
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
DATA5FUNC	DATA4FUNC	DATA3FUNC	DATA2FUNC	DATA1FUNC	DATA0FUNC	CLKFUNC	SYNCFUNC
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 29-23. DMM Pin Control 0 (DMMPC0) Field Descriptions**

Bit	Field	Value	Description
31-19	Reserved	0	Reads returns 0. Writes have no effect.
18	ENAFUNC	0 1 0 1	<p><b>Functional mode of DMMENA pin.</b> This bit defines whether the pin is used in functional mode or in GIO mode.</p> <p>User and privilege mode (read):</p> <p>0 Pin is used in GIO mode 1 Pin is used in Functional mode</p> <p>User and privilege mode (write):</p> <p>0 Pin is used in GIO mode 1 Pin is used in Functional mode</p>
17-2	DATAxFUNC	0 1 0 1	<p><b>Functional mode of DMMDATA[x] pin.</b> This bit defines whether the pin is used in functional mode or in GIO mode. If pins are configured in functional mode, only pins defined in Table 29-5 have to be used for proper operation.</p> <p>User and privilege mode (read):</p> <p>0 Pin is used in GIO mode 1 Pin is used in Functional mode</p> <p>User and privilege mode (write):</p> <p>0 Pin is used in GIO mode 1 Pin is used in Functional mode</p>
1	CLKFUNC	0 1 0 1	<p><b>Functional mode of DMMCLK pin.</b> This bit defines whether the pin is used in functional mode or in GIO mode.</p> <p>User and privilege mode (read):</p> <p>0 Pin is used in GIO mode 1 Pin is used in Functional mode</p> <p>User and privilege mode (write):</p> <p>0 Pin is used in GIO mode 1 Pin is used in Functional mode</p>

**Table 29-23. DMM Pin Control 0 (DMMPC0) Field Descriptions (continued)**

Bit	Field	Value	Description
0	SYNCFUNC		<b>Functional mode of DMMSYNC pin.</b> This bit defines whether the pin is used in functional mode or in GIO mode.
			User and privilege mode (read):
		0	Pin is used in GIO mode
		1	Pin is used in Functional mode
			User and privilege mode (write):
		0	Pin is used in GIO mode
		1	Pin is used in Functional mode

### 29.3.17 DMM Pin Control 1 (DMMPC1)

The bits in this register define the direction of the individual module pins when in GIO mode.

**Figure 29-23. DMM Pin Control 1 (DMMPC1) [offset = 70h]**

31	30	29	28	27	26	25	24
Reserved							
R-0							
23	22	21	20	19	18	17	16
Reserved					ENADIR	DATA15DIR	DATA14DIR
R-0					R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
DATA13DIR	DATA12DIR	DATA11DIR	DATA10DIR	DATA9DIR	DATA8DIR	DATA7DIR	DATA6DIR
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
DATA5DIR	DATA4DIR	DATA3DIR	DATA2DIR	DATA1DIR	DATA0DIR	CLKDIR	SYNCDIR
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 29-24. DMM Pin Control 1 (DMMPC1) Field Descriptions**

Bit	Field	Value	Description
31-19	Reserved	0	Reads returns 0. Writes have no effect.
18	ENADIR	0 1 0 1	<b>Direction of DMMENA pin.</b> User and privilege mode (read): Pin is used as input Pin is used as output User and privilege mode (write): Pin is set to input Pin is set to output
17-2	DATAxDIR	0 1 0 1	<b>Direction of DMMDATA[x] pin.</b> This bit defines whether the pin is used as input or output in GIO mode. User and privilege mode (read): Pin is used as input Pin is used as output User and privilege mode (write): Pin is set to input Pin is set to output
1	CLKDIR	0 1 0 1	<b>Direction of DMMCLK pin.</b> This bit defines whether the pin is used as input or output in GIO mode. User and privilege mode (read): Pin is used as input Pin is used as output User and privilege mode (write): Pin is set to input Pin is set to output

**Table 29-24. DMM Pin Control 1 (DMMPC1) Field Descriptions (continued)**

Bit	Field	Value	Description
0	SYNCDIR		<b>Direction of DMMSYNC pin.</b> This bit defines whether the pin is used as input or output in GIO mode.
			User and privilege mode (read):
		0	Pin is used as input
		1	Pin is used as output
			User and privilege mode (write):
		0	Pin is set to input
		1	Pin is set to output

### 29.3.18 DMM Pin Control 2 (DMMP2)

The bits in this register reflect the digital representation of the voltage level at the module pins. Even if a pin is configured to be an output pin, the level can be read back via this register.

**Figure 29-24. DMM Pin Control 2 (DMMP2) [offset = 74h]**

31	30	29	28	27	26	25	24
Reserved							
R-0							
23	22	21	20	19	18	17	16
Reserved				ENAIN		DATA15IN	DATA14IN
R-0				R/W-0		R/W-0	R/W-0
15	14	13	12	11	10	9	8
DATA13IN	DATA12IN	DATA11IN	DATA10IN	DATA9IN	DATA8IN	DATA7IN	DATA6IN
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
DATA5IN	DATA4IN	DATA3IN	DATA2IN	DATA1IN	DATA0IN	CLKIN	SYNCIN
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 29-25. DMM Pin Control 2 (DMMP2) Field Descriptions**

Bit	Field	Value	Description
31-19	Reserved	0	Reads returns 0. Writes have no effect.
18	ENAIN	0 1	<b>DMMEN<math>\bar{A}</math> input.</b> This bit reflects the state of the pin in all modes. User and privilege mode (read): Logic low (input voltage is $V_{IL}$ or lower) Logic high (input voltage is $V_{IH}$ or higher) Privilege mode (write): writes to this bit have no effect.
17-2	DATA $x$ IN	0 1	<b>DMMDATA<math>[x]</math> input.</b> This bit reflects the state of the pin in all modes. User and privilege mode (read): Logic low (input voltage is $V_{IL}$ or lower) Logic high (input voltage is $V_{IH}$ or higher) Privilege mode (write): writes to this bit have no effect.
1	CLKIN	0 1	<b>DMMCLK input.</b> This bit reflects the state of the pin in all modes. User and privilege mode (read): Logic low (input voltage is $V_{IL}$ or lower) Logic high (input voltage is $V_{IH}$ or higher) Privilege mode (write): writes to this bit have no effect.
0	SYNCIN	0 1	<b>DMMSYNC input.</b> This bit reflects the state of the pin in all modes. User and privilege mode (read): Logic low (input voltage is $V_{IL}$ or lower) Logic high (input voltage is $V_{IH}$ or higher) Privilege mode (write): writes to this bit have no effect.

**29.3.19 DMM Pin Control 3 (DMMP3)**

The bits in this register set the pin to logic low or high level if the pin is configured as output (Section 29.3.17).

**Figure 29-25. DMM Pin Control 3 (DMMP3) [offset = 78h]**

31	30	29	28	27	26	25	24
Reserved							
R-0							
23	22	21	20	19	18	17	16
Reserved					ENAOUT	DATA15OUT	DATA14OUT
R-0					R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
DATA13OUT	DATA12OUT	DATA11OUT	DATA10OUT	DATA9OUT	DATA8OUT	DATA7OUT	DATA6OUT
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
DATA5OUT	DATA4OUT	DATA3OUT	DATA2OUT	DATA1OUT	DATA0OUT	CLKOUT	SYNCOUT
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 29-26. DMM Pin Control 3 (DMMP3) Field Descriptions**

Bit	Field	Value	Description
31-19	Reserved	0	Reads returns 0. Writes have no effect.
18	ENAOUT	0 1 0 1	<b>Output state of DMMENA pin.</b> This bit sets the pin to logic low or high level User and privilege mode (read): Logic low (output voltage is $V_{OL}$ or lower) Logic high (output voltage is $V_{OH}$ or higher) User and privilege mode (write): Logic low (output voltage is set to $V_{OL}$ or lower) Logic high (output voltage is set to $V_{OH}$ or higher)
17-2	DATAxOUT	0 1 0 1	<b>Output state of DMMDATA[x] pin.</b> This bit sets the pin to logic low or high level. User and privilege mode (read): Logic low (output voltage is $V_{OL}$ or lower) Logic high (output voltage is $V_{OH}$ or higher) User and privilege mode (write): Logic low (output voltage is set to $V_{OL}$ or lower) Logic high (output voltage is set to $V_{OH}$ or higher)
1	CLKOUT	0 1 0 1	<b>Output state of DMMCLK pin.</b> This bit sets the pin to logic low or high level User and privilege mode (read): Logic low (output voltage is $V_{OL}$ or lower) Logic high (output voltage is $V_{OH}$ or higher) User and privilege mode (write): Logic low (output voltage is set to $V_{OL}$ or lower) Logic high (output voltage is set to $V_{OH}$ or higher)



**Table 29-26. DMM Pin Control 3 (DMMPC3) Field Descriptions (continued)**

Bit	Field	Value	Description
0	SYNCOUT		<b>Output state of DMMSYNC pin.</b> This bit sets the pin to logic low or high level.
			User and privilege mode (read):
		0	Logic low (output voltage is $V_{OL}$ or lower)
		1	Logic high (output voltage is $V_{OH}$ or higher)
			User and privilege mode (write):
		0	Logic low (output voltage is set to $V_{OL}$ or lower)
		1	Logic high (output voltage is set to $V_{OH}$ or higher)

**29.3.20 DMM Pin Control 4 (DMMPC4)**

This register allows to set individual pins to a logic high level without having to do a read-modify-write operation as would be the case with the DMMPC3 register (Section 29.3.19). Writing a zero to a bit will not change the state of the pin.

**Figure 29-26. DMM Pin Control 4 (DMMPC4) [offset = 7Ch]**

31	30	29	28	27	26	25	24
Reserved							
R-0							
23	22	21	20	19	18	17	16
Reserved				ENASET		DATA15SET	DATA14SET
R-0				R/W-0		R/W-0	R/W-0
15	14	13	12	11	10	9	8
DATA13SET	DATA12SET	DATA11SET	DATA10SET	DATA9SET	DATA8SET	DATA7SET	DATA6SET
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
DATA5SET	DATA4SET	DATA3SET	DATA2SET	DATA1SET	DATA0SET	CLKSET	SYNCSET
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 29-27. DMM Pin Control 4 (DMMPC4) Field Descriptions**

Bit	Field	Value	Description
31-19	Reserved	0	Reads returns 0. Writes have no effect.
18	ENASET	0	Logic low (output voltage is $V_{OL}$ or lower)
		1	Logic high (output voltage is $V_{OH}$ or higher)
		0	State of the pin is unchanged
		1	Logic high (output voltage is set to $V_{OH}$ or higher)
17-2	DATAxSET	0	Logic low (output voltage is $V_{OL}$ or lower)
		1	Logic high (output voltage is $V_{OH}$ or higher)
		0	State of the pin is unchanged
		1	Logic high (output voltage is set to $V_{OH}$ or higher)
1	CLKSET	0	Logic low (output voltage is $V_{OL}$ or lower)
		1	Logic high (output voltage is $V_{OH}$ or higher)
		0	State of the pin is unchanged
		1	Logic high (output voltage is set to $V_{OH}$ or higher)

**Table 29-27. DMM Pin Control 4 (DMMP4) Field Descriptions (continued)**

Bit	Field	Value	Description
0	SYNCSET		<b>Sets output state of DMMSYNC pin logic high.</b> Value in the SYNCSET bit sets the data output control register bit to 1 regardless of the current value in the SYNCOUT bit.
			User and privilege mode (read):
		0	Logic low (output voltage is $V_{OL}$ or lower)
		1	Logic high (output voltage is $V_{OH}$ or higher)
			User and privilege mode (write):
		0	State of the pin is unchanged
		1	Logic high (output voltage is set to $V_{OH}$ or higher)

**29.3.21 DMM Pin Control 5 (DMMP5)**

This register allows to set individual pins to a logic low level without having to do a read-modify-write operation as would be the case with the DMMP3 register (Section 29.3.19). Writing a one to a bit will change the output to a logic low level, writing a zero will not change the state of the pin.

**Figure 29-27. DMM Pin Control 5 (DMMP5) [offset = 80h]**

31	30	29	28	27	26	25	24
Reserved							
R-0							
23	22	21	20	19	18	17	16
Reserved					ENACL	DATA15CLR	DATA14CLR
R-0					R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
DATA13CLR	DATA12CLR	DATA11CLR	DATA10CLR	DATA9CLR	DATA8CLR	DATA7CLR	DATA6CLR
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
DATA5CLR	DATA4CLR	DATA3CLR	DATA2CLR	DATA1CLR	DATA0CLR	CLKCLR	SYNCLR
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 29-28. DMM Pin Control 5 (DMMP5) Field Descriptions**

Bit	Field	Value	Description
31-19	Reserved	0	Reads returns 0. Writes have no effect.
18	ENACL	0 1 0 1	<p><b>Sets output state of DMMENA pin to logic low.</b> Value in the ENACL bit clears the data output control register bit to 0 regardless of the current value in the ENAOUT bit.</p> <p>User and privilege mode (read):</p> <p>Logic low (output voltage is <math>V_{OL}</math> or lower)</p> <p>Logic high (output voltage is <math>V_{OH}</math> or higher)</p> <p>User and privilege mode (write):</p> <p>0: State of the pin is unchanged</p> <p>1: Clears the pin to logic low (output voltage is set to <math>V_{OL}</math> or lower)</p>
17-2	DATAxCLR	0 1 0 1	<p><b>Sets output state of DMMDATA[x] pin to logic low.</b> Value in the DATAxCLR bit clears the data output control register bit to 0 regardless of the current value in the DATAxOUT bit.</p> <p>User and privilege mode (read):</p> <p>Logic low (output voltage is <math>V_{OL}</math> or lower)</p> <p>Logic high (output voltage is <math>V_{OH}</math> or higher)</p> <p>User and privilege mode (write):</p> <p>0: State of the pin is unchanged</p> <p>1: Clears the pin to logic low (output voltage is set to <math>V_{OL}</math> or lower)</p>
1	CLKCLR	0 1 0 1	<p><b>Sets output state of DMMCLK pin to logic low.</b> Value in the CLKCLR bit clears the data output control register bit to 0 regardless of the current value in the DATAxOUT bit.</p> <p>User and privilege mode (read):</p> <p>Logic low (output voltage is <math>V_{OL}</math> or lower)</p> <p>Logic high (output voltage is <math>V_{OH}</math> or higher)</p> <p>User and privilege mode (write):</p> <p>0: State of the pin is unchanged</p> <p>1: Clears the pin to logic low (output voltage is set to <math>V_{OL}</math> or lower)</p>

**Table 29-28. DMM Pin Control 5 (DMMP5) Field Descriptions (continued)**

Bit	Field	Value	Description
0	SYNCCLR		<b>Sets output state of DMMSYNC pin to logic low.</b> Value in the SYNCCLR bit clears the data output control register bit to 0 regardless of the current value in the DATAxOUT bit.
			User and privilege mode (read):
		0	Logic low (output voltage is $V_{OL}$ or lower)
		1	Logic high (output voltage is $V_{OH}$ or higher)
			User and privilege mode (write):
		0	State of the pin is unchanged
		1	Clears the pin to logic low (output voltage is set to $V_{OL}$ or lower)

**29.3.22 DMM Pin Control 6 (DMMPC6)**

These bits configure the pins in push-pull or open-drain functionality. If configured to be open-drain, the module only drives a logic low level on the pin. An external pull-up resistor needs to be connected to the pin to pull it high when the pin is in high-impedance mode.

**Figure 29-28. DMM Pin Control 6 (DMMPC6) [offset = 84h]**

31	30	29	28	27	26	25	24
Reserved							
R-0							
23	22	21	20	19	18	17	16
Reserved				ENAPDR		DATA15PDR	DATA14PDR
R-0				R/W-0		R/W-0	R/W-0
15	14	13	12	11	10	9	8
DATA13PDR	DATA12PDR	DATA11PDR	DATA10PDR	DATA9PDR	DATA8PDR	DATA7PDR	DATA6PDR
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
DATA5PDR	DATA4PDR	DATA3PDR	DATA2PDR	DATA1PDR	DATA0PDR	CLKPDR	SYNCPDR
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 29-29. DMM Pin Control 6 (DMMPC6) Field Descriptions**

Bit	Field	Value	Description
31-19	Reserved	0	Reads returns 0. Writes have no effect.
18	ENAPDR	0 1	<p><b>Open Drain enable.</b> Enables open drain functionality if the pin is configured as GIO output (DMMPC0[18]=0; DMMPC1[18]=1). If the pin is configured as functional pin (DMMPC0[18]=1), the open drain functionality is disabled.</p> <p>User and privilege mode (read):</p> <p>0 Pin behaves as normal push/pull pin</p> <p>1 Pin operates in open drain mode</p> <p>User and privilege mode (write):</p> <p>0 Configures pin as push/pull</p> <p>1 Configures pin as open drain</p>
17-2	DATAxPDR	0 1	<p><b>Open Drain enable.</b> Enables open drain functionality on pin if pin is configured as GIO output (DMMPC0[x]=0; DMMPC1[x]=1). If the pin is configured as functional pin (DMMPC0[x] = 1), the open drain functionality is disabled.</p> <p>User and privilege mode (read):</p> <p>0 Pin behaves as normal push/pull pin</p> <p>1 Pin operates in open drain mode</p> <p>User and privilege mode (write):</p> <p>0 Configures the pin as push/pull</p> <p>1 Configures the pin as open drain</p>
1	CLKPDR	0 1	<p><b>Open Drain enable.</b> Enables open drain functionality on pin if pin is configured as GIO output (DMMPC0[1]=0; DMMPC1[1]=1). If the pin is configured as functional pin (DMMPC0[1] = 1), the open drain functionality is disabled.</p> <p>User and privilege mode (read):</p> <p>0 Pin behaves as normal push/pull pin</p> <p>1 Pin operates in open drain mode</p> <p>User and privilege mode (write):</p> <p>0 Configures the pin as push/pull</p> <p>1 Configures the pin as open drain</p>

**Table 29-29. DMM Pin Control 6 (DMMPC6) Field Descriptions (continued)**

Bit	Field	Value	Description
0	SYNCPDR		<p><b>Open Drain enable.</b> Enables open drain functionality on pin if pin is configured as GIO output (DMMPC0[0]=0; DMMPC1[0]=1). If the pin is configured as functional pin (DMMPC0[0] = 1), the open drain functionality is disabled.</p> <p>User and privilege mode (read):</p> <p>0 Pin behaves as normal push/pull pin</p> <p>1 Pin operates in open drain mode</p> <p>User and privilege mode (write):</p> <p>0 Configures the pin as push/pull</p> <p>1 Configures the pin as open drain</p>

**29.3.23 DMM Pin Control 7 (DMMPC7)**

The bits in register control the pullup/down functionality of a pin. The internal pullup/down can be enabled or disabled by this register. The reset configuration of these bits is device implementation dependent. Please consult the device datasheet this information.

**Figure 29-29. DMM Pin Control 7 (DMMPC7) [offset = 88h]**

31	30	29	28	27	26	25	24
Reserved							
R-0							
23	22	21	20	19	18	17	16
Reserved				ENAPDIS		DATA15PDIS	DATA14PDIS
R-0				R/W-x		R/W-x	R/W-x
15	14	13	12	11	10	9	8
DATA13PDIS	DATA12PDIS	DATA11PDIS	DATA10PDIS	DATA9PDIS	DATA8PDIS	DATA7PDIS	DATA6PDIS
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
7	6	5	4	3	2	1	0
DATA5PDIS	DATA4PDIS	DATA3PDIS	DATA2PDIS	DATA1PDIS	DATA0PDIS	CLKPDIS	SYNCPDIS
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 29-30. DMM Pin Control 7 (DMMPC7) Field Descriptions**

Bit	Field	Value	Description
31-19	Reserved	0	Reads returns 0. Writes have no effect.
18	ENAPDIS	0	<b>Pull disable.</b> Removes internal pullup/pulldown functionality from pin when configured as input pin (DMMPC1[18] = 0). User and privilege mode (read): Pullup/pulldown functionality enabled
		1	Pullup/pulldown functionality disabled
		0	User and privilege mode (write): Enables pullup/pulldown functionality
		1	Disables pullup/pulldown functionality
17-2	DATAxPDIS	0	<b>Pull disable.</b> Removes internal pullup/pulldown functionality from pin when configured as input pin (DMMPC1[x] = 0). User and privilege mode (read): Pullup/pulldown functionality enabled
		1	Pullup/pulldown functionality disabled
		0	User and privilege mode (write): Enables pullup/pulldown functionality
		1	Disables pullup/pulldown functionality
1	CLKPDIS	0	<b>Pull disable.</b> Removes internal pullup/pulldown functionality from pin when configured as input pin (DMMPC1[1] = 0). User and privilege mode (read): Pullup/pulldown functionality enabled
		1	Pullup/pulldown functionality disabled
		0	User and privilege mode (write): Enables pullup/pulldown functionality
		1	Disables pullup/pulldown functionality



**Table 29-30. DMM Pin Control 7 (DMMPC7) Field Descriptions (continued)**

Bit	Field	Value	Description
0	SYNCPDIS		<b>Pull disable.</b> Removes internal pullup/pulldown functionality from pin when configured as input pin (DMMPC1[0] = 0).
			User and privilege mode (read):
		0	Pullup/pulldown functionality enabled
		1	Pullup/pulldown functionality disabled
			User and privilege mode (write):
		0	Enables pullup/pulldown functionality
		1	Disables pullup/pulldown functionality

---

**NOTE:** If the pullup/down is disabled in DMMPC7 and configured as pulldown in DMMPC8 ([Section 29.3.24](#)), then the input buffer is disabled.

---

**29.3.24 DMM Pin Control 8 (DMMPC8)**

These bits control if the internal pullup or pulldown is configured on the input pin. A secondary function exists when the pull configuration is disabled and a pulldown is selected. This will disable the input buffer.

**Figure 29-30. DMM Pin Control 8 (DMMPC8) [offset = 8Ch]**

31	30	29	28	27	26	25	24
Reserved							
R-0							
23	22	21	20	19	18	17	16
Reserved					ENAPSEL	DATA15PSEL	DATA14PSEL
R-0					R/W-1	R/W-1	R/W-1
15	14	13	12	11	10	9	8
DATA13PSEL	DATA12PSEL	DATA11PSEL	DATA10PSEL	DATA9PSEL	DATA8PSEL	DATA7PSEL	DATA6PSEL
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
7	6	5	4	3	2	1	0
DATA5PSEL	DATA4PSEL	DATA3PSEL	DATA2PSEL	DATA1PSEL	DATA0PSEL	CLKPSEL	SYNCPSEL
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 29-31. DMM Pin Control 8 (DMMPC8) Field Descriptions**

Bit	Field	Value	Description
31-19	Reserved	0	Reads returns 0. Writes have no effect.
18	ENAPSEL	0 1	<b>Pull select.</b> Configures pullup or pulldown functionality if DMMPC7[18] = 0). User and privilege mode (read): 0 Pulldown functionality enabled 1 Pullup functionality enabled User and privilege mode (write): 0 Enables pulldown functionality 1 Enables pullup functionality
17-2	DATAxPSEL	0 1	<b>Pull select.</b> Configures pullup or pulldown functionality if DMMPC7[x] = 0). User and privilege mode (read): 0 Pulldown functionality enabled 1 Pullup functionality enabled User and privilege mode (write): 0 Enables pulldown functionality 1 Enables pullup functionality
1	CLKPSEL	0 1	<b>Pull select.</b> Configures pullup or pulldown functionality if DMMPC7[1] = 0). User and privilege mode (read): 0 Pulldown functionality enabled 1 Pullup functionality enabled User and privilege mode (write): 0 Enables pulldown functionality 1 Enables pullup functionality

**Table 29-31. DMM Pin Control 8 (DMMPC8) Field Descriptions (continued)**

Bit	Field	Value	Description
0	SYNCPSEL		<b>Pull select.</b> Configures pullup or pulldown functionality if DMMPC7[0] = 0).
		0	User and privilege mode (read): Pulldown functionality enabled
		1	Pullup functionality enabled
			User and privilege mode (write):
		0	Enables pulldown functionality
		1	Enables pullup functionality

---

**NOTE:** If the pullup/down is disabled in DMMPC7 ([Section 29.3.23](#)) and configured as pulldown in DMMPC8, then the input buffer is disabled.

---

## **RAM Trace Port (RTP)**

---

---

---

This chapter describes the functionality of the RAM trace port (RTP) module. It allows the capability to do data trace of CPU or other master accesses to the internal RAM and peripherals.

<b>Topic</b>	<b>Page</b>
<b>30.1 Overview</b> .....	<b>1866</b>
<b>30.2 Module Operation</b> .....	<b>1867</b>
<b>30.3 GIO Function</b> .....	<b>1873</b>
<b>30.4 Control Registers</b> .....	<b>1874</b>

## 30.1 Overview

This document describes the functionality of the RAM trace port (RTP) module, which provides the features to datalog the RAM contents of the devices or accesses to peripherals without program intrusion. It can trace all data write or read accesses to internal RAM. In addition, it provides the capability to directly transfer data to a FIFO to support a CPU-controlled transmission of the data. The trace data is transmitted over a dedicated external interface.

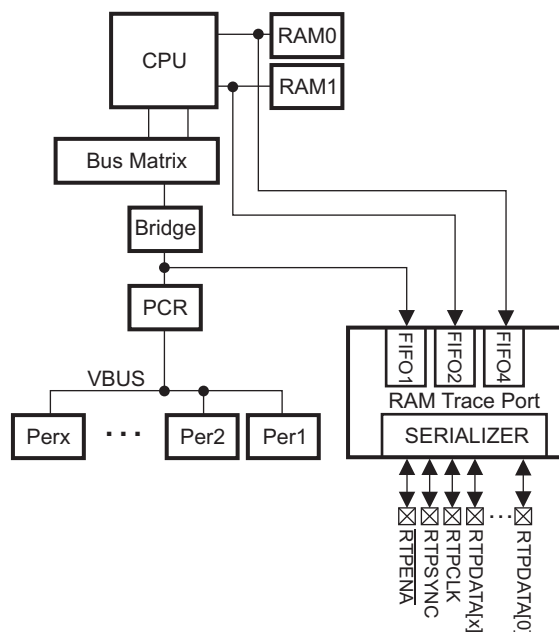
### 30.1.1 Features

The RTP offers the following features:

- Two modes of operation - Trace Mode and Direct Data Mode
  - Trace Mode ([Section 30.2.1](#))
    - Non-intrusive data trace on write or read operation
    - Visibility of RAM content at any time on external capture hardware
    - Trace of peripheral accesses
    - 2 configurable trace regions for each RAM module to limit amount of data to be traced
    - FIFO to store data and address of data of multiple read/write operations
    - Trace of CPU and/or DMA accesses with indication of the master in the transmitted data packet
  - Direct Data Mode ([Section 30.2.2](#))
    - Directly write data with the CPU or trace read operations to a FIFO, without transmitting header and address information
- Dedicated synchronous interface to transmit data to external devices
- Free-running clock generation or clock stop mode between transmissions
- up to 100 Mbit per sec/pin transfer rate for transmitting data (up to 133MB/s; see device datasheet for maximum transmission clock frequency)
- Pins not used in functional mode can be used as GIOs

### 30.1.2 Block Diagram

**Figure 30-1. Block Diagram RAM Trace Port Module**



## 30.2 Module Operation

The RTP module has two modes of operation: Trace mode and Direct Data Mode.

### 30.2.1 Trace Mode

This mode traces all write or read accesses of CPU and/or a different master to the internal RAMs and the peripheral bus, if the access falls into one of the programmed trace regions. The trace regions allow to restrict the amount of data which is traced. This is done by specifying the start address and the size of the region to be traced. It is not possible to trace write and read operations in the same region at the same time.

Whenever a write or read access occurs, the address, data, size of the access (8, 16, 32, 64 bit), and which module initiated the write or read operation is captured into the FIFO of the corresponding RAM frame. Once new data is in the FIFO and the serializer is empty, the RTP transmits the data into the serializer and starts transmitting it.

The FIFOs are shifting data into the serializer in a round-robin scheme. This means if data is available in multiple FIFOs, the sequence for shifting data into the serializer is FIFO1, FIFO2 and then FIFO4. Only one entry in the respective FIFO is provided to the serializer before switching to the next FIFO. If a FIFO does not hold new data, it will be skipped. This scheme ensures that the FIFOs are drained uniformly.

#### 30.2.1.1 Packet Format in Trace Mode

Figure 30-2 and Figure 30-3 illustrate this format.

Figure 30-2. Packet Format Trace Mode for RAM Locations

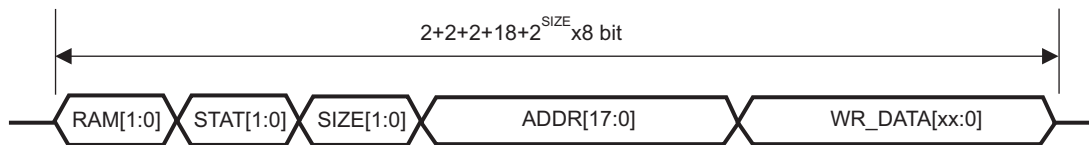
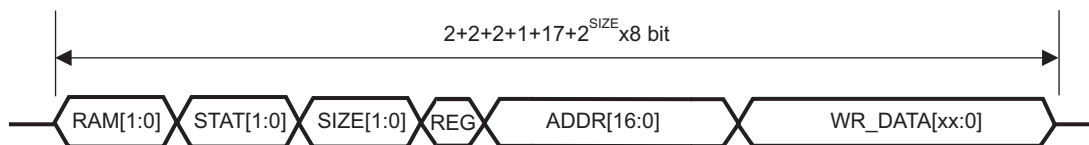


Figure 30-3. Packet Format Trace Mode for Peripheral Locations



When RAM locations are traced, one packet consists of two bits denoting the RAM block in which the data is stored or if the access has been to a peripheral location (Table 30-1), two status bits showing the access initiator or if there was a FIFO overflow (Table 30-2), two bit size (8, 16, 32, or 64 bit) information of the data (Table 30-3), the 18-bit address for RAM accesses and  $2^{SIZE} \times 8$  bits of data. If a peripheral location is traced, then the effective address reduces to 17 bits (ADDR[16:0]) and a separate bit (REG) between the SIZE information and the address denotes which programmable region has traced this peripheral access (Table 30-4). With the region identifier, the external hardware can determine which peripheral was traced.

Table 30-1. Encoding of RAM Bits in Trace Mode Packet Format

RAM[1:0]	RAM
0	Cortex-R4 B0TCM
1h	Cortex-R4 B1TCM
2h	Reserved
3h	Peripherals

**Table 30-2. Encoding of Status Bits in Trace Mode Packet Format**

STAT[1:0]	Status
0	normal entry CPU
1h	normal entry other Master
2h	reserved
3h	overflow of the dedicated FIFO

In the event of a FIFO overflow, an overflow will be signaled in the status bits of the next transmitted packet of that particular FIFO. The last entry in the FIFO will not be overwritten by the new data.

**Table 30-3. Encoding of SIZE bits in Trace Mode Packet Format**

SIZE[1:0]	Write/Read Size
0	8 bit
1h	16 bit
2h	32 bit
3h	64 bit

**Table 30-4. Encoding of REG in Trace Mode Packet Format**

REG	Region
0	1
1	2

The packet will be split up into several subpackets when transmitted over the RTP port pins depending on the port width configured. The port width is configured with bits PW[1:0] in the RTPGLBCTRL register (Section 30.4.1). For certain port width configurations and write/read sizes, the number of bits in a packet does not exactly match the port width for the last subpacket. The remaining bits will be filled with zeros.

**Table 30-5. Number of Transfers/Packet**

Port Width	Write/Read Size in bits			
	8	16	32	64
2	16 --> 16	20 --> 20	28 --> 28	44 --> 44
4	8 --> 8	10 --> 10	14 --> 14	22 --> 22
8	4 --> 4	5 --> 5	7 --> 7	11 --> 11
16	2 --> 2	2.5 --> 3	3.5 --> 4	5.5 --> 6

Example: For a 16-bit port and with data of 16-bit, the last transfer has to be padded with eight 0s. This effectively results in a transfer of 48 bits instead of 40. However the whole transfer is completed in 3 RTPCLK cycles.

For a detailed description of the representation of the packet on the RTP port pins, please refer to Section 30.2.5.

### 30.2.2 Direct Data Mode (DDM)

In this mode, data is written directly by the CPU or other master to a dedicated capture register, RTPDDMW (Section 30.4.7). The data is then transferred from the capture register to the FIFO. In a different configuration the module traces the data on read operations on the RAM directly into the FIFOs. In Direct Data Mode, no information other than the actual data is transmitted. The address of the written data can only be determined by the order of writes or reads by the CPU or other master. This mode is especially useful if a block of data on consecutive addresses has to be transmitted.

The transfer size (8, 16, or 32 bit) is programmable, but cannot be dynamically changed. Data not

written/read in the correct transfer size will be truncated/extended. For example, if the transfer size is programmed to 16 bits and a 32-bit write operation is performed, the data written to the FIFO will be 32-bit wide, however only the upper 16 bits of the FIFO will be transmitted. If an 8-bit operation is performed, bits 8-15 of the FIFO will be indeterminate, so the upper 8 bits of the data transmitted are dependent on the previous contents of the FIFO RAM.

When the module is configured in Direct Data Mode (TM\_DDM = 1) to trace write operations (DDM\_RW = 1) to the RTPDDMW register, the programming of the trace regions for all FIFOs will be ignored and data tracing, when accessing the addresses defined by the regions, will not occur. If the module is configured in read mode (DDM\_RW = 0), and if the read access to a RAM block falls into a valid trace region, the data will be traced into the corresponding FIFO for this RAM block. Since no address information is transmitted in Direct Data Mode, the executing program has to make sure that one FIFO is completely empty (RTPGSR, Section 30.4.3), before new data is traced into the next FIFO.

---

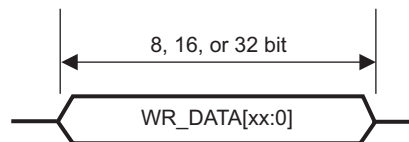
**NOTE:** Direct Data Mode read operation is not supported on devices with a Cortex-R4 CPU, due to the bus protocol of the TCM interface and certain performance enhancements (e.g. data packing) implemented in the core.

---

### 30.2.2.1 Packet Format in Direct Data Mode

Figure 30-4 illustrates this format.

**Figure 30-4. Packet Format in Direct Data Mode**



In Direct Data Mode write or read operations, only the data written to the RTPDDMW register (Section 30.4.7) or the data read from RAM, and therefore captured into the FIFO, will be transmitted. The packet length is programmable (8, 16, or 32 bits).

### 30.2.3 Trace Regions

To limit the amount of data to be trace, two trace regions per RAM or peripheral are implemented. These can be programmed to specific start addresses and block sizes. Depending on the device configuration (number of RAM blocks), not all regions might be implemented. Trace regions are used in Trace Mode for read or write trace and in Direct Data Mode for read trace. In Direct Data Mode write configuration, the data has to be written directly to RTPDDMW (Section 30.4.7).

The RAM and peripherals start at fixed addresses in the devices memory map. With this the start address of a region does not need to be specified with its full 32-bit address. For RAM regions, only the lower 18-bit need to be programmed. The peripheral address frame covers a wider range and the start address needs to be programmed with the lower 24-bit.

The trace regions do not support a programmable end address; however, a block size needs to be specified for each region. The block size can be chosen from as low as 256 Bytes up to 256 kBytes (128 kBytes for peripherals).

#### 30.2.3.1 Inverse Trace Regions

The RTP can be configured to trace accesses which fall into, or are made outside of the specified regions. This can be accomplished by the INV\_RGN bit. If this bit is 0, all access which are made inside a region are traced. If the bit is 1, all accesses outside the region are traced. The INV\_RGN bit affects all regions of the RTP (Section 30.4.1).

There are certain restrictions when using INV\_RGN = 1:

- In this mode up to 2 regions can be excluded from tracing accesses to a particular RAM.



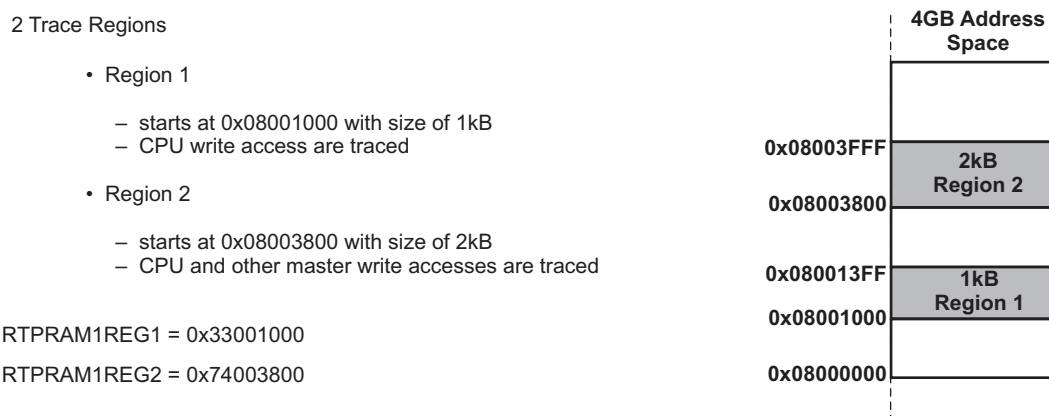
- Inverse trace regions with one or both regions of a RAM programmed with blocksize = 0 is not supported. If only one address range should be excluded from the trace, either the address range has to be covered by both regions (e.g. excluding 1kB range with two 512B regions), or both regions have to be programmed with the same start address and region size. If the whole RAM should be traced, inverse region mode should not be used, instead the 2 regions could be programmed to cover the entire address range with INV\_RGN = 0.
- Both regions have to define the same access rights (bits CPU\_DMA and RW; see [Section 30.4.4](#)) for accesses outside of the region of each RAM block, otherwise the result is undefined.
- Peripheral trace in inverse region mode is not supported. The 16 MByte peripheral address range cannot be covered entirely by the 17 bit address definition of the RTP protocol.

### 30.2.3.2 Overlapping Trace Regions

When in INV\_RGN = 0 mode with both regions overlapping and an access is done into the overlapping address range, both regions will be checked for their access rights and if one or both is satisfied, the access will be traced. In the case that both regions would allow the data to be traced, there will still be only one entry into the FIFO.

If accesses to peripherals are done within overlapping regions, the REG bit in the protocol will be 0, denoting Region 1 ([Section 30.2.1.1](#)).

**Figure 30-5. Example for Trace Region Setup**



### 30.2.3.3 Cortex-R4 Specifics

Due to the bus system used on Cortex-R4, special considerations have to be taken into account. Block diagram RAM Trace Port Module shows the block diagram of the RTP connected to the Cortex-R4 RAM interface.

Both interfaces to RAM0 and RAM1 are 64-bit wide. RAM0 and RAM1 are building a consecutive address range, where the 64-bit addresses 00, 10h, 20h, ... reside in RAM0 and 08h, 18h, 28h, ... reside in RAM1.

#### Considerations/Restrictions

- To trace a certain address range, the regions of both RAM0 (RTPRAM1REG[1:2], [Section 30.4.4](#)) and RAM1 (RTPRAM2REG[1:2], [Section 30.4.5](#)) need to be set up for the same start address and region size. Otherwise only every other 64-bit access will be traced.
- Direct Data Mode read operation is not supported on Cortex-R4. This is because the CPU uses 64-bit accesses for read operations even though the intended accesses is sub-64-bit wide. Since Direct Data Mode only supports 32-bit data transfers, and the Cortex-R4 RAM interface does not provide information which byte out of the 64-bit is accessed with the read operation, the RTP cannot determine the correct data value.
- If the RAM is protected by ECC, only 64-bit write accesses can be traced. Every 64-bit word in the RAM is protected by a corresponding 8-bit ECC checksum. When a sub-64-bit write access is performed, the Cortex-R4 has to do a read-modify-write operation of the 64-bit word to be modified in order to calculate the corresponding checksum and then write it back to memory. External hardware

can still determine which portion of the 64-bit word has been modified, since the other bytes in this word did not change.

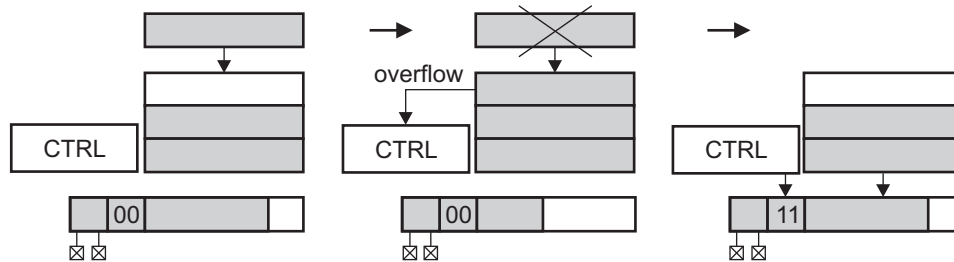
### 30.2.4 Overflow/Empty Handling

In case the application does RAM accesses faster than the FIFO can be emptied via the external pin interface, the FIFO can overflow. The user can choose whether the program execution/data transfer should be suspended, or an overflow should be signaled in the status bits of the next, to be transmitted, message of this particular FIFO. If program execution is resumed, the data will be lost. The overflow will not be signaled in the message that is already in the serializer and being transmitted when the overflow occurs.

**NOTE:** The status information will only be transmitted in Trace Mode, since the Direct Data Mode packet does not contain any status information.

When an overflow in a FIFO occurs, the corresponding bit in RTPGSR (Section 30.4.3) will be set.

Figure 30-6. FIFO Overflow Handling



### 30.2.5 Signal Description

RTPCLK	This clock signal is used to clock out the data of the serializer. Depending on the CONTCLK bit, the clock can be suspended between packets or it can be free running. The RTPCLK frequency can be adjusted by the PRESCALER bits (Section 30.4.1).
RTPSYNC	The module provides a packet-sync signal. This signal will go high on the rising edge of RTPCLK and will be valid for one RTPCLK cycle to synchronize external hardware to the data stream. The RTPSYNC pulse will be generated for each new packet.
$\overline{\text{RTPENA}}$	This signal is an input and can be used by external hardware to stop the data transmission between packets. When the $\overline{\text{RTPENA}}$ signal goes high, the RTP will finish the current packet transmission and then stop. Once the signal is pulled low again, the RTP will resume the transfer if data is still present in the serializer or FIFOs. The $\overline{\text{RTPENA}}$ signal does not have to be used for proper module operation. It can be used in GIO mode if the external hardware cannot generate this signal. Overflows of the external system cannot be handled in this case.
RTPDATA[15:0]	These pins are used to do the actual data transfer. Data changes with the rising edge of RTPCLK. The port can be configured for different widths (PW[1:0]). The minimum port width supported is 2 pins. See Table 30-9 which pins are used for the port.

Figure 30-7 shows an example of multiple packet transmissions in Trace Mode with an interruption between packets because of  $\overline{\text{RTPENA}}$  pulled high.

**Figure 30-7. RTP Packet Transfer with Sync Signal**

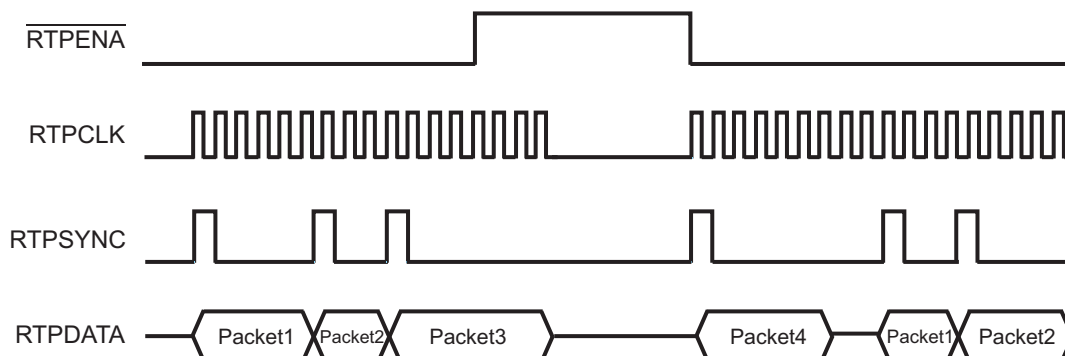
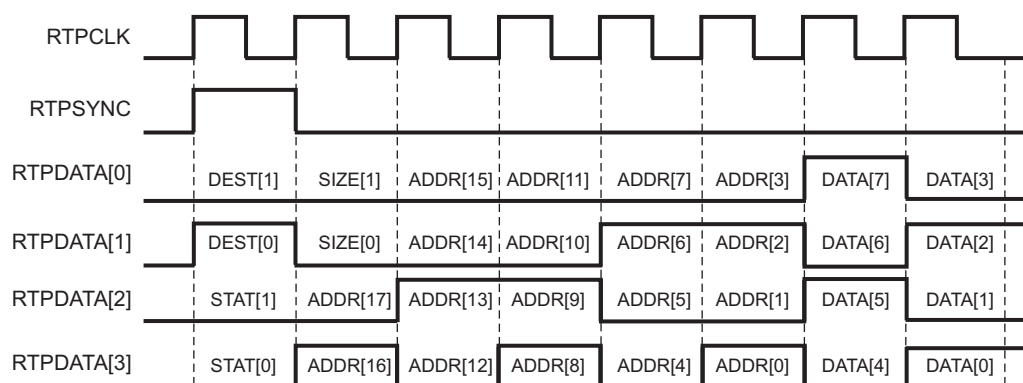


Figure 30-8 shows an example of a 4-bit data port with 8-bit write data (A5h) written into RAM1 (address 12345h) with no overflow in trace mode.

**Figure 30-8. Packet Format in Trace Mode**



### 30.2.6 Data Rate

The module is configurable to support different RTPCLK frequencies. Please see the device datasheet for the maximum supported frequency. HCLK will be prescaled to achieve the desired RTPCLK frequency. The prescaler supports prescale values from 1 to 8 (RTPGLBCTRL, [Section 30.4.1](#)).

The effective bandwidth depends on the configuration of the module and the average data width transmitted in the packets.

### 30.3 GIO Function

Pins which are not used for RTP functionality can be used as normal GIO pins. If pins should be used in functional mode or GIO mode can be programmed in RTPPC0 ([Section 30.4.8](#)). The direction of the pins can be chosen in RTPPC1 ([Section 30.4.9](#)).

Module pins can have either an internal pullup or active pulldown that makes it possible to leave the pins unconnected externally when configured as inputs. The pins can be programmed to have the active pull capability by writing a 0 to the corresponding bit in the RTPPC7 register ([Section 30.4.15](#)). Writing a 1 to the corresponding bit disables the active pull functionality of the pin. A pull up can be configured by writing 1 to the corresponding bit in RTPPC8 register ([Section 30.4.16](#)). Writing 0 will activate the pulldown capability. The pullup/pulldown is deactivated when a bidirectional pin is configured as an output. If the pullup/down capability is disabled (RTPPC7) and the pull is configured as pulldown (RTPPC8) the input buffer will be disabled.

The GIO pin can be configured to include an open drain functionality when they are configured as output pins. This is done by writing a 1 into the corresponding bit of the RTPPC6 register ([Section 30.4.14](#)).

When the open drain functionality is enabled, a zero written to the data output register (RTPPC3, [Section 30.4.11](#)) forces the pin to a low output voltage (VOL or lower), whereas writing a 1 to the data output register (RTPPC3) forces the pin to a high impedance state. The open drain functionality is disabled when the pin is configured as an input pin.

## 30.4 Control Registers

This section describes the RTP module registers. The registers support 8-bit, 16-bit, and 32-bit writes. The base address of the RTP module is FFFF FA00h.

**Table 30-6. Module Registers**

Offset	Acronym	Register Description	Section
00	RTPGLBCTRL	RTP Global Control Register	<a href="#">Section 30.4.1</a>
04h	RTPTRENA	RTP Trace Enable Register	<a href="#">Section 30.4.2</a>
08h	RTPGSR	RTP Global Status Register	<a href="#">Section 30.4.3</a>
0Ch, 10h	RTPRAM1REG	RTP RAM 1 Trace Region Register	<a href="#">Section 30.4.4</a>
14h, 18h	RTPRAM2REG	RTP RAM 2 Trace Region Register	<a href="#">Section 30.4.5</a>
24h, 28h	RTPPERREG	RTP Peripheral Trace Region Register	<a href="#">Section 30.4.6</a>
2Ch	RTPDDMW	RTP Direct Data Mode Write Register	<a href="#">Section 30.4.7</a>
34h	RTPPC0	RTP Pin Control 0 Register	<a href="#">Section 30.4.8</a>
38h	RTPPC1	RTP Pin Control 1 Register	<a href="#">Section 30.4.9</a>
3Ch	RTPPC2	RTP Pin Control 2 Register	<a href="#">Section 30.4.10</a>
40h	RTPPC3	RTP Pin Control 3 Register	<a href="#">Section 30.4.11</a>
44h	RTPPC4	RTP Pin Control 4 Register	<a href="#">Section 30.4.12</a>
48h	RTPPC5	RTP Pin Control 5 Register	<a href="#">Section 30.4.13</a>
4Ch	RTPPC6	RTP Pin Control 6 Register	<a href="#">Section 30.4.14</a>
50h	RTPPC7	RTP Pin Control 7 Register	<a href="#">Section 30.4.15</a>
54h	RTPPC8	RTP Pin Control 8 Register	<a href="#">Section 30.4.16</a>

### 30.4.1 RTP Global Control Register (RTPGLBCTRL)

The configuration of the module can be changed with this register. [Figure 30-9](#) and [Table 30-7](#) illustrate this register.

**Figure 30-9. RTP Global Control Register (RTPGLBCTRL) [offset = 00h]**

31	25	24	23	19	18	16	
Reserved		TEST	Reserved		PRESCALER		
R-0		R/W-0	R-0		R/W-7h		
15	14	13	12	11	10	9	
Reserved		DDM_WIDTH		DDM_RW	TM_DDM	PW	
R-0		R/W-0		R/W-0	R/W-0	R/W-0	
7	6	5	4	3	0		
RESET	CONTCLK	HOVF	INV_RGN	ON/OFF			
R/W-0	R/W-0	R/W-0	R/W-0	R/W-5h			

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 30-7. RTP Global Control Register (RTPGLBCTRL) Field Descriptions**

Bit	Field	Value	Description
31-25	Reserved	0	Read returns 0. Writes have no effect.
24	TEST	0	By setting the bit, the FIFO RAM will be mapped into the SYSTEM Peripheral frame starting at address 0xFFF83000. Each FIFO will start at a 1k boundary. Each FIFO entry will be aligned to a 128 bit boundary. See <a href="#">Table 30-8</a> for a listing of the FIFOs and their corresponding addresses.  User and privilege mode (read): The FIFO RAM is not accessible in the memory map. The FIFO RAM is mapped to address FFF8 3000h  Privilege mode (write): Disables mapping of the FIFO RAM Enables mapping of the FIFO RAM into address FFF8 3000h
		1	
		0	
		1	
18-16	PRESCALER	0	The prescaler divides HCLK down to the desired RTPCLK frequency. The maximum RTPCLK frequency specified in the device datasheet must not be exceeded. No dynamic change of RTPCLK is supported. The module should be switched off by the ON/OFF bits in this register before changing the prescaler.  User and privilege mode read, privilege mode write: The prescaler is HCLK/1. The prescaler is HCLK/2. The prescaler is HCLK/3. The prescaler is HCLK/4. The prescaler is HCLK/5. The prescaler is HCLK/6. The prescaler is HCLK/7. The prescaler is HCLK/8.
		1h	
		2h	
		3h	
		4h	
		5h	
		6h	
		7h	
15-14	Reserved	0	Read returns 0. Writes have no effect.
13-12	DDM_WIDTH	0	Direct data mode word size width. This bit field configures the number of bits that will be transmitted in Direct Data Mode.  User and privilege mode read, privilege mode write: The word size width is 8 bits. The word size width is 16 bits. The word size width is 32 bits. Reserved
		1h	
		2h	
		3h	

**Table 30-7. RTP Global Control Register (RTPGLBCTRL) Field Descriptions (continued)**

Bit	Field	Value	Description
11	DDM_RW	<p>0 1</p> <p>0 1</p>	<p>Direct Data Mode</p> <p>User and privilege mode (read):</p> <p>Read tracing in Direct Data Mode is enabled.</p> <p>Write tracing in Direct Data Mode to DDMW register is enabled.</p> <p>Privilege mode (write):</p> <p>0 1</p> <p>Enable read tracing in Direct Data Mode. The RW bits in the RTPRAMxREGy registers to be ignored.</p> <p>Write tracing in Direct Data Mode to DDMW register is enabled. The RW bits in the RTPRAMxREGy registers to be ignored (<a href="#">Section 30.4.4</a>).</p>
10	TM_DDM	<p>0 1</p> <p>0 1</p>	<p>Trace Mode or Direct Data Mode</p> <p>User and privilege mode (read):</p> <p>Module is configured in Trace Mode.</p> <p>Module is configured in Direct Data Mode.</p> <p>Privilege mode (write):</p> <p>0 1</p> <p>Configure module to Trace Mode.</p> <p>Configure module to Direct Data Mode.</p>
9-8	PW	<p>0 1h 2h 3h</p>	<p>Port width. This bit field configures the RTP to the desired port width. Pins that are not used for functional mode can be used as GIO pins. See <a href="#">Table 30-9</a> which pins are used for the port.</p> <p>The RTP is 2 pins wide.</p> <p>The RTP is 4 pins wide.</p> <p>The RTP is 8 pins wide.</p> <p>The RTP is 16 pins wide.</p>
7	RESET	<p>0 1</p> <p>0 1</p>	<p>This bit resets the state machine and the registers to their reset value. This reset ensures that no data left in the FIFOs is shifted out after switching on the module with the ON/OFF bit.</p> <p>User and privilege mode (read):</p> <p>The RTP module is out of reset.</p> <p>The RTP module is in reset.</p> <p>Privilege mode (write):</p> <p>0 1</p> <p>Do not reset the module.</p> <p>Reset the module.</p>
6	CONTCLK	<p>0 1</p> <p>0 1</p>	<p>Continuous RTPCLK enable.</p> <p>User and privilege mode (read):</p> <p>The RTPCLK is stopped between transmissions.</p> <p>The RTPCLK is free running.</p> <p>Privilege mode (write):</p> <p>0 1</p> <p>Stop RTPCLK between transmissions.</p> <p>Configure RTPCLK as free running.</p>
5	HOVF	<p>0 1</p> <p>0 1</p>	<p>Halt on overflow. This bit indicates whether the CPU or DMA is halted while only one location in the FIFO is empty in Trace Mode or Direct Data Mode (read).</p> <p>User and privilege mode (read):</p> <p>The current data transfer to the FIFO will not be suspended in case of a full FIFO.</p> <p>The current data transfer to the FIFO will be suspended in case of a full FIFO.</p> <p>Privilege mode (write):</p> <p>0 1</p> <p>The halt on FIFO overflow will be disabled. The data transfer will not be suspended and will be discarded. Data written to the RTPDDMW register will overwrite the RTPDDMW register (<a href="#">Section 30.4.7</a>).</p> <p>The halt on FIFO overflow will be enabled. Data written to the already full FIFO will be written once the FIFO is emptied again. The data transfer to the FIFO will be suspended and signaled to the CPU or other master while there is still data to be shifted out. When there is an empty FIFO location again, the transfer of the data to the FIFO will be finished.</p>

**Table 30-7. RTP Global Control Register (RTPGLBCTRL) Field Descriptions (continued)**

Bit	Field	Value	Description
4	INV_RGN		Trace inside or outside of defined trace regions.
		0	User and privilege mode (read): Accesses inside the trace regions are traced.
		1	Accesses outside the trace regions are traced.
			Privilege mode (write):
	0	Allow tracing of accesses inside the regions set in RTPRAMxREGy	
	1	Allow tracing of accesses outside the regions set in RTPRAMxREGy ( <a href="#">Section 30.4.4</a> )	
3-0	ON/OFF		ON/Off switch.
			User and privilege mode (read):
		Ah	Tracing of data is enabled.
		All Others	Tracing of data is disabled.
		Privilege mode (write):	
	Ah	Enable Tracing of data. If there is any previous captured data remaining, it will be shifted out.	
	All Others	Disable tracing of data. If there is still data left in the shift register, it will be shifted out before disabling the shift operations. The data captured in the FIFO remains there until the ON/OFF bits are set to Ah.	
		<b>NOTE:</b> It is recommended to write 5h to disable the module to prevent a soft error from enabling the module inadvertently by a single bit flip.	

**Table 30-8. FIFO Corresponding Addresses**

FIFO	Address
1	FFF8 3000h
2	FFF8 3400h
4	FFF8 3C00h

**Table 30-9. Pins Used for Data Communication**

Port Width (PW)	Pins Used
00	RTPDATA[1:0]
01	RTPDATA[3:0]
10	RTPDATA[7:0]
11	RTPDATA[15:0]



### 30.4.2 RTP Trace Enable Register (RTPTRENA)

This register enables/disables tracing of the different RAM blocks or the peripherals individually. [Figure 30-10](#) and [Table 30-10](#) illustrate this register.

**Figure 30-10. RTP Trace Enable Register (RTPTRENA) [offset = 04h]**

31	25	24	23	16
Reserved	ENA4		Reserved	
R-0	R/W-0		R-0	
15	9	8	7	1 0
Reserved	ENA2		Reserved	
R-0	R/W-0		R-0	ENA1
				R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 30-10. RTP Trace Enable Register (RTPTRENA) Field Descriptions**

Bit	Field	Value	Description
31-25	Reserved	0	Read returns 0. Writes have no effect.
24	ENA4	0	Enable tracing for peripherals. This bit enables tracing into FIFO4 in trace mode (read/write) or direct data mode (read) operations. In Direct Data Mode write operations, this bit will be ignored and tracing into FIFO4 will be disabled. User and privilege mode (read): Tracing is disabled.
		1	Tracing is enabled.
		0	Privilege mode (write): Disable tracing. If RTPGLBCTRL.ON/OFF = Ah, data already captured in FIFO4 will still be transmitted ( <a href="#">Section 30.4.1</a> ).
		1	Enable tracing.
23-9	Reserved	0	Read returns 0. Writes have no effect.
8	ENA2	0	Enable tracing for RAM block 2. This bit enables tracing into FIFO2 in Trace Mode (read/write) or Direct Data Mode (read) operations. In Direct Data Mode write operations, this bit will be ignored and tracing into FIFO2 will be disabled. User and privilege mode (read): Tracing is disabled.
		1	Tracing is enabled.
		0	Privilege mode (write): Disable tracing. If RTPGLBCTRL.ON/OFF = Ah, data already captured in FIFO2 will still be transmitted.
		1	Enable tracing.
7-1	Reserved	0	Read returns 0. Writes have no effect.
0	ENA1	0	Enable tracing for RAM block 1. This bit enables tracing into FIFO1 in Trace Mode (read/write) or Direct Data Mode (read) operations. In Direct Data Mode write operations, this bit will be ignored and tracing into FIFO1 will be disabled. User and privilege mode (read): Tracing is disabled.
		1	Tracing is enabled.
		0	Privilege mode (write): Disable tracing. If RTPGLBCTRL.ON/OFF = Ah, data already captured in FIFO1 will still be transmitted.
		1	Enable tracing.



**Table 30-11. RTP Global Status Register (RTPGSR) [offset = 08h] Field Descriptions (continued)**

Bit	Field	Value	Description
1	OVF2		<p><b>Overflow RAM block 2 FIFO.</b> This flag indicates that FIFO2 had all locations full and another attempt to write data to it occurred. The bit will not be cleared automatically if the FIFO is emptied again. The bit will stay set until the CPU clears it.</p> <p>User and privilege mode (read):</p> <p>0 No overflow occurred.</p> <p>1 An overflow occurred.</p> <p>Privilege mode (write):</p> <p>0 Writing a zero to this bit has no effect.</p> <p>1 The bit is cleared.</p>
0	OVF1		<p><b>Overflow RAM block 1 FIFO.</b> This flag indicates that FIFO1 had all locations full and another attempt to write data to it occurred. The bit will not be cleared automatically if the FIFO is emptied again. The bit will stay set until the CPU clears it.</p> <p>User and privilege mode (read):</p> <p>0 No overflow occurred.</p> <p>1 An overflow occurred.</p> <p>Privilege mode (write):</p> <p>0 Writing a zero to this bit has no effect.</p> <p>1 The bit is cleared.</p>

### 30.4.4 RTP RAM 1 Trace Region [1:2] Register (RTPRAM1REG[1:2])

Figure 30-12 and Table 30-12 illustrate these registers.

**Figure 30-12. RTP RAM 1 Trace Region [1:2] Register (RTPRAM1REG[1:2]) [offset = 0Ch, 10h]**

31	30	29	28	27	24	23	18	17	16
Rsvd.	CPU_DMA	RW	BLOCKSIZE			Reserved		STARTADDR	
R-0	R/W-0	R/W-0	R/W-0			R-0		R/W-0	
15									0
STARTADDR									
R/W-0									

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 30-12. RTP RAM 1 Trace Region [1:2] Register (RTPRAM1REG[1:2]) Field Descriptions**

Bit	Field	Value	Description
31	Reserved	0	Read returns 0. Writes have no effect.
30-29	CPU_DMA	0 1h 2h 3h	<p><b>CPU and/or other master access.</b> This bit field indicates if read or write operations are traced either coming from the CPU and/or from the other master.</p> <p>User and privilege mode read, privilege mode write:</p> <p>0 Read or write operations are traced when coming from the CPU and the other master.</p> <p>1h Read or write operations are traced only when coming from the CPU.</p> <p>2h Read or write operations are traced only when coming from the other master.</p> <p>3h Reserved</p>
28	RW	0 1 0 1	<p><b>Read/Write.</b> This bit indicates if read or write operations are traced in Trace Mode or Direct Data Mode (read operation). If configured for write in Direct Data Mode (RTPGLBCTRL, <a href="#">Section 30.4.1</a>), the data captured will be discarded. A write operation in Direct Data Mode has to be directly to the RTPDDMW register (<a href="#">Section 30.4.7</a>) instead of to RAM. Depending on the INV_RGN bit setting, accesses into or outside the region will be traced.</p> <p>User and privilege mode (read):</p> <p>0 Read operations will be captured.</p> <p>1 Write operations will be captured.</p> <p>Privilege mode (write):</p> <p>0 Trace read accesses.</p> <p>1 Trace write accesses.</p>
27-24	BLOCKSIZE	0 1h 2h 3h 4h Ah Bh Ch-Fh	<p>These bits define the length of the trace region. Depending on the setting of INV_RGN (<a href="#">Section 30.4.1</a>), accesses inside or outside the region defined by the start address and blocksize will be traced. If all bits of BLOCKSIZE are 0, the region is disabled and no data will be captured.</p> <p>Region size (in bytes):</p> <p>0 0</p> <p>1h 256</p> <p>2h 512</p> <p>3h 1K</p> <p>4h 2K</p> <p>Ah 128K</p> <p>Bh 256K</p> <p>Ch-Fh Reserved</p>
23-18	Reserved	0	Read returns 0. Writes have no effect.
17-0	STARTADDR	0-3 FFFFh	These bits define the starting address of the address region that should be traced. The start address has to be a multiple of the block size chosen. If the start address is not a multiple of the block size, the start of the region will begin at the next lower block size boundary.

### 30.4.5 RTP RAM 2 Trace Region [1:2] Register (RTPRAM2REG[1:2])

Figure 30-13 and Table 30-13 illustrate these registers.

**Figure 30-13. RTP RAM 2 Trace Region [1:2] Register (RTPRAM2REG[1:2]) [offset = 14h, 18h]**

31	30	29	28	27	24	23	18	17	16
Rsvd.	CPU_DMA	RW	BLOCKSIZE			Reserved			STARTADDR
R-0	R/W-0	R/W-0	R/W-0			R-0			R/W-0
15									0
STARTADDR									
R/W-0									

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 30-13. RTP RAM 2 Trace Region [1:2] Register (RTPRAM2REG[1:2]) Field Descriptions**

Bit	Field	Value	Description
31	Reserved	0	Read returns 0. Writes have no effect.
30-29	CPU_DMA	0 1h 2h 3h	<p><b>CPU and/or other master access.</b> This bit field indicates if read or write operations are traced either coming from the CPU and/or from the other master.</p> <p>User and privilege mode read, privilege mode write:</p> <p>0 Read or write operations are traced when coming from the CPU and the other master.</p> <p>1h Read or write operations are traced only when coming from the CPU.</p> <p>2h Read or write operations are traced only when coming from the other master.</p> <p>3h Reserved</p>
28	RW	0 1 0 1	<p><b>Read/Write.</b> This bit indicates if read or write operations are traced in Trace Mode or Direct Data Mode (read operation). If configured for write in Direct Data Mode (RTPGLBCTRL, <a href="#">Section 30.4.1</a>), the data captured will be discarded. A write operation in Direct Data Mode has to be directly to the RTPDDMW register (<a href="#">Section 30.4.7</a>) instead of to RAM. Depending on the INV_RGN bit setting, accesses into or outside the region will be traced.</p> <p>User and privilege mode (read):</p> <p>0 Read operations will be captured.</p> <p>1 Write operations will be captured.</p> <p>Privilege mode (write):</p> <p>0 Trace read accesses.</p> <p>1 Trace write accesses.</p>
27-24	BLOCKSIZE	0 1h 2h 3h 4h Ah Bh Ch-Fh	<p>These bits define the length of the trace region. Depending on the setting of INV_RGN (<a href="#">Section 30.4.1</a>), accesses inside or outside the region defined by the start address and blocksize will be traced. If all bits of BLOCKSIZE are 0, the region is disabled and no data will be captured.</p> <p>Region size (in bytes):</p> <p>0 0</p> <p>1h 256</p> <p>2h 512</p> <p>3h 1K</p> <p>4h 2K</p> <p>Ah 128K</p> <p>Bh 256K</p> <p>Ch-Fh Reserved</p>
23-18	Reserved	0	Read returns 0. Writes have no effect.
17-0	STARTADDR	0-3 FFFFh	These bits define the starting address of the address region that should be traced. The start address has to be a multiple of the block size chosen. If the start address is not a multiple of the block size, the start of the region will begin at the next lower block size boundary.

### 30.4.6 RTP Peripheral Trace Region [1:2] Registers (RTPPERREG[1:2])

FIFO4 is dedicated for tracing the peripheral accesses. Since the peripheral frame is 16 Mbytes, the start address has to be defined as a 24-bit value. However, only bits 16 to 0 will be transmitted in the protocol. Bit REG (Section 30.2.1.1) in the protocol will be 0 if there was an access to the range defined by RTPPERREG1. REG will be 1 if the access was into the range defined by RTPPERREG2. Figure 30-14 and Table 30-14 illustrate these registers.

**Figure 30-14. RTP Peripheral Trace Region [1:2] Register (RTPPERREG[1:2]) [offset = 24h, 28h]**

31	30	29	28	27	24	23	16
Rsvd.	CPU_DMA	RW	BLOCKSIZE			STARTADDR	
R-0	R/W-0	R/W-0	R/W-0			R/W-0	
15							0
STARTADDR							
R/W-0							

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

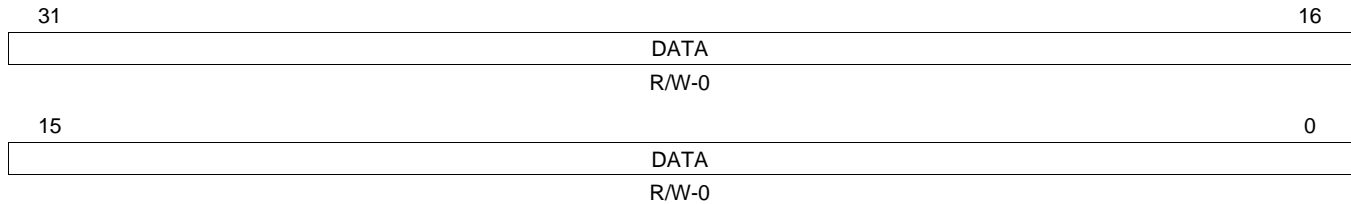
**Table 30-14. RTP Peripheral Trace Region [1:2] Register (RTPPERREG[1:2]) Field Descriptions**

Bit	Field	Value	Description
31	Reserved	0	Read returns 0. Writes have no effect.
30-29	CPU_DMA	0 1h 2h 3h	<p><b>CPU and/or other master access.</b> This bit field indicates if read or write operations are traced either coming from the CPU and/or from the other master.</p> <p>User and privilege mode read, privilege mode write:</p> <p>0 Read or write operations are traced when coming from the CPU and the other master.</p> <p>1h Read or write operations are traced only when coming from the CPU.</p> <p>2h Read or write operations are traced only when coming from the other master.</p> <p>3h Reserved</p>
28	RW	0 1 0 1	<p><b>Read/Write.</b> This bit indicates if read or write operations are traced in Trace Mode or Direct Data Mode (read operation). If configured for write in Direct Data Mode (RTPGLBCTRL, Section 30.4.1), the data captured will be discarded. A write operation in Direct Data Mode has to be directly to the RTPDDMW register (Section 30.4.7) instead of to RAM. Depending on the INV_RGN bit setting, accesses into or outside the region will be traced.</p> <p>User and privilege mode (read):</p> <p>0 Read operations will be captured.</p> <p>1 Write operations will be captured.</p> <p>Privilege mode (write):</p> <p>0 Trace read accesses.</p> <p>1 Trace write accesses.</p>
27-24	BLOCKSIZE	0 1h 2h 3h 4h Ah Bh Ch-Fh	<p>These bits define the length of the trace region. Depending on the setting of INV_RGN (Section 30.4.1), accesses inside or outside the region defined by the start address and blocksize will be traced. If all bits of BLOCKSIZE are 0, the region is disabled and no data will be captured.</p> <p>Region size (in bytes):</p> <p>0 0</p> <p>1h 256</p> <p>2h 512</p> <p>3h 1K</p> <p>4h 2K</p> <p>Ah 128K</p> <p>Bh 256K</p> <p>Ch-Fh Reserved</p>
23-0	STARTADDR	0-FF FFFFh	These bits define the starting address of the address region that should be traced. The start address has to be a multiple of the block size chosen. If the start address is not a multiple of the block size, the start of the region will begin at the next lower block size boundary.

### 30.4.7 RTP Direct Data Mode Write Register (RTPDDMW)

The CPU has to write data to this register if the module is used in Direct Data Mode write configuration. [Figure 30-15](#) and [Table 30-15](#) illustrate this register.

**Figure 30-15. RTP Direct Data Mode Write Register (RTPDDMW) [offset = 2Ch]**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

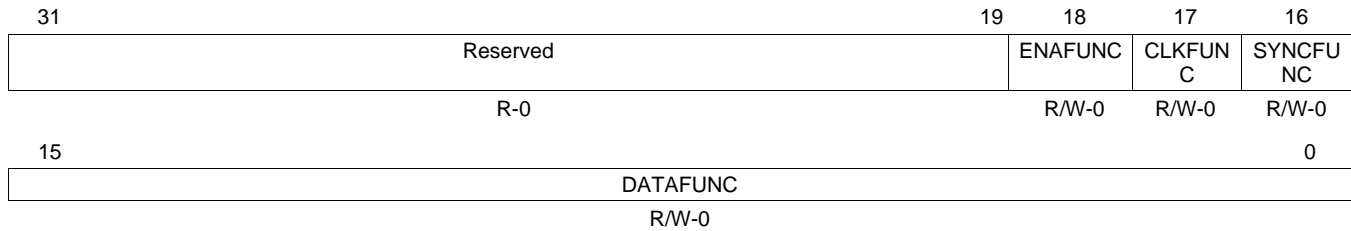
**Table 30-15. RTP Direct Data Mode Write Register (RTPDDMW) [offset = 2Ch] Field Descriptions**

Bit	Field	Value	Description
31-0	DATA	0-FFFF FFFFh	This register must be written to in a Direct Data Mode write operation to store the data into FIFO1. Data written must be right-aligned. If the FIFO is full, the reaction depends on the setting of the HOVF bit ( <a href="#">Section 30.4.1</a> ). If the bit is set, the master writing the data will be waitstated. If the bit is cleared, previous data written to the register will be overwritten. Reads of this register always return 0.

### 30.4.8 RTP Pin Control 0 Register (RTPPC0)

This register configures the RTP pins as functional or GIO pins. Once the pin is configured in functional mode, it overrides the settings in the RTPPC1 register (Section 30.4.9). Writing to RTPPC3, RTPPC4 and RTPPC5 will have no effect for pins configured as functional pins. Figure 30-16 and Table 30-16 illustrate this register.

**Figure 30-16. RTP Pin Control 0 Register (RTPPC0) [offset = 34h]**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 30-16. RTP Pin Control 0 Register (RTPPC0) [offset = 34h] Field Descriptions**

Bit	Field	Value	Description
31-19	Reserved	0	Read returns 0. Writes have no effect.
18	ENAFUNC		<b>Functional mode of RTPEN<math>\bar{A}</math> pin.</b>
		0	User and privilege mode (read): Pin is used in GIO mode.
		1	Pin is used in functional mode.
			User and privilege mode (write):
17	CLKFUNC		<b>Functional mode of RTPCLK pin.</b>
		0	User and privilege mode (read): Pin is used in GIO mode.
		1	Pin is used in functional mode.
			User and privilege mode (write):
16	SYNCFUNC		<b>Functional mode of RTPSYNC pin.</b>
		0	User and privilege mode (read): Pin is used in GIO mode.
		1	Pin is used in functional mode.
			User and privilege mode (write):
15-0	DATAFUNC		<b>Functional mode of RTPDATA[15:0] pins.</b> These bits define whether the pins are used in functional mode or in GIO mode. Each bit represents a single pin.
		0	User and privilege mode (read): Pin is used in GIO mode.
		1	Pin is used in functional mode.
			User and privilege mode (write):
15-0	DATAFUNC		
		0	Configure pin to GIO mode.
		1	Configure pin to functional mode.



### 30.4.9 RTP Pin Control 1 Register (RTPPC1)

Once the pin is configured in functional mode (RTPPC0, [Section 30.4.8](#)), configuring the corresponding bit in RTPPC1 to 0 will not disable the output driver. [Figure 30-17](#) and [Table 30-17](#) illustrate this register.

**Figure 30-17. RTP Pin Control 1 Register (RTPPC1) [offset = 38h]**

31	Reserved	19	ENADIR	18	CLKDIR	17	SYNCDIR	16
	R-0		R/W-0		R/W-0		R/W-0	
15	DATADIR						0	
	R/W-0							

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 30-17. RTP Pin Control 1 Register (RTPPC1) Field Descriptions**

Bit	Field	Value	Description
31-19	Reserved	0	Read returns 0. Writes have no effect.
18	ENADIR		<b>Direction of RTPENA pin.</b> This bit defines whether the pin is used as input or output in GIO mode. This bit has no effect when the pin is configured in functional mode.
			User and privilege mode (read):
		0	Pin is used as input.
		1	Pin is used as output.
17	CLKDIR		<b>Direction of RTPCLK pin.</b> This bit defines whether the pin is used as input or output in GIO mode. This bit has no effect when the pin is configured in functional mode.
			User and privilege mode (read):
		0	Pin is used as input.
		1	Pin is used as output.
16	SYNCDIR		<b>Direction of RTPSYNC pin.</b> This bit defines whether the pin is used as input or output in GIO mode. This bit has no effect when the pin is configured in functional mode.
			User and privilege mode (read):
		0	Pin is used as input.
		1	Pin is used as output.
15-0	DATADIR		<b>Direction of RTPDATA[15:0] pins.</b> These bits define whether the pins are used as input or output in GIO mode. These bits have no effect when the pins are configured in functional mode. Each bit represents a single pin.
			User and privilege mode (read):
		0	Pin is used as input.
		1	Pin is used as output.
			User and privilege mode (write):
		0	Configure pin to input mode.
		1	Configure pin to output mode.

### 30.4.10 RTP Pin Control 2 Register (RTPPC2)

This register represents the input value of the pins if when in GIO or functional mode. [Figure 30-18](#) and [Table 30-18](#) illustrate this register.

**Figure 30-18. RTP Pin Control 2 Register (RTPPC2) [offset = 3Ch]**

31	Reserved	19	ENAIN	18	CLKIN	17	SYNCIN	16
	R-0		R-x		R-x		R-x	
15	DATAIN						0	
	R-x							

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 30-18. RTP Pin Control 2 Register (RTPPC2) [offset = 3Ch] Field Descriptions**

Bit	Field	Value	Description
31-19	Reserved	0	Read returns 0. Writes have no effect.
18	ENAIN	0 1	<b>RTPEN<math>\bar{A}</math> input.</b> This bit reflects the state of the pin in all modes. Writes to this bit have no effect. User and privilege mode (read): The pin is at logic low (0) (input voltage is $V_{IL}$ or lower). The pin is at logic high (1) (input voltage is $V_{IH}$ or higher).
17	CLKIN	0 1	<b>RTPCLK input.</b> This bit reflects the state of the pin in all modes. Writes to this bit have no effect. User and privilege mode (read): The pin is at logic low (0) (input voltage is $V_{IL}$ or lower). The pin is at logic high (1) (input voltage is $V_{IH}$ or higher).
16	SYNCIN	0 1	<b>RTPSYNC input.</b> This bit reflects the state of the pin in all modes. Writes to this bit have no effect. User and privilege mode (read): The pin is at logic low (0) (input voltage is $V_{IL}$ or lower). The pin is at logic high (1) (input voltage is $V_{IH}$ or higher).
15-0	DATAIN	0 1	<b>RTPDATA[15:0] input.</b> These bits reflect the state of the pins in all modes. Each bit represents a single pin. Writes to this bit have no effect. User and privilege mode (read): The pin is at logic low (0) (input voltage is $V_{IL}$ or lower). The pin is at logic high (1) (input voltage is $V_{IH}$ or higher).

### 30.4.11 RTP Pin Control 3 Register (RTPPC3)

The bits in this register define the state of the pins when configured in GIO mode as output pins. Once a pin is configured in functional mode (RTPPC0, [Section 30.4.8](#)), changing the state of the corresponding bit in RTPPC3 will not affect the pin's state. [Figure 30-19](#) and [Table 30-19](#) illustrate this register.

**Figure 30-19. RTP Pin Control 3 Register (RTPPC3) [offset = 40h]**

31	Reserved	19	18	17	16
			ENAOUT	CLKOUT	SYNCOU T
R-0			R/W-0	R/W-0	R/W-0
15	DATAOUT				0
R/W-0					

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 30-19. RTP Pin Control 3 Register (RTPPC3) Field Descriptions**

Bit	Field	Value	Description
31-19	Reserved	0	Read returns 0. Writes have no effect.
18	ENAOUT	0 1 0 1	<b>RTPENA output. This pin sets the output state of the RTPENA pin.</b> User and privilege mode (read): The pin is configured to output logic low (0) (output voltage is $V_{OL}$ or lower). The pin is configured to output logic high (1) (output voltage is $V_{OH}$ or higher). User and privilege mode (write): Set pin to logic low (0) (output voltage is $V_{OL}$ or lower). Set pin to logic high (1) (output voltage is $V_{OH}$ or higher).
17	CLKOUT	0 1 0 1	<b>RTPCLK output. This pin sets the output state of the RTPCLK pin.</b> User and privilege mode (read): The pin is configured to output logic low (0) (output voltage is $V_{OL}$ or lower). The pin is configured to output logic high (1) (output voltage is $V_{OH}$ or higher). User and privilege mode (write): Set pin to logic low (0) (output voltage is $V_{OL}$ or lower). Set pin to logic high (1) (output voltage is $V_{OH}$ or higher).
16	SYNCOU T	0 1 0 1	<b>RTPSYNC output. This pin sets the output state of the RTPSYNC pin.</b> User and privilege mode (read): The pin is configured to output logic low (0) (output voltage is $V_{OL}$ or lower). The pin is configured to output logic high (1) (output voltage is $V_{OH}$ or higher). User and privilege mode (write): Set pin to logic low (0) (output voltage is $V_{OL}$ or lower). Set pin to logic high (1) (output voltage is $V_{OH}$ or higher).
15-0	DATAOUT	0 1 0 1	<b>RTPDATA[15:0] output. These bits set the output state of the RTPDATA[15:0] pins. Each bit represents a single pin.</b> User and privilege mode (read): The pin is configured to output logic low (0) (output voltage is $V_{OL}$ or lower). The pin is configured to output logic high (1) (output voltage is $V_{OH}$ or higher). User and privilege mode (write): Set pin to logic low (0) (output voltage is $V_{OL}$ or lower). Set pin to logic high (1) (output voltage is $V_{OH}$ or higher).

### 30.4.12 RTP Pin Control 4 Register (RTPPC4)

This register provides the option to set pins to a logic 1 level without influencing the state of other pins. It eliminates the read-modify-write operation necessary with RTPPC3 (Section 30.4.11). Once the pin is configured in functional mode (RTPPC0, Section 30.4.8), setting the corresponding bit to one in RTPPC4 will not affect the pin's state. Figure 30-20 and Table 30-20 illustrate this register.

**Figure 30-20. RTP Pin Control 4 Register (RTPPC4) [offset = 44h]**

31	Reserved	19	18	17	16
			ENASET	CLKSET	SYNCSET
R-0			R/W-0	R/W-0	R/W-0
15	DATASET				0
R/W-0					

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 30-20. RTP Pin Control 4 Register (RTPPC4) Field Descriptions**

Bit	Field	Value	Description
31-19	Reserved	0	Read returns 0. Writes have no effect.
18	ENASET	0 1	<p><b>Sets the output state of RTPENA pin to logic high.</b> Value in the ENASET bit sets the data output control register bit to 1 regardless of the current value in the ENAOUT bit .</p> <p>User and privilege mode (read):</p> <p>0 The pin is configured to output a logic low (0) (output voltage is <math>V_{OL}</math> or lower).</p> <p>1 The pin is configured to output logic high (1) (output voltage is <math>V_{OH}</math> or higher).</p> <p>User and privilege mode (write):</p> <p>0 Writing a zero to this bit has no effect.</p> <p>1 Set pin to logic high (1) (output voltage is <math>V_{OH}</math> or higher).</p>
17	CLKSET	0 1	<p><b>Sets the output state of RTPCLK pin to logic high.</b> Value in the CLKSET bit sets the data output control register bit to 1 regardless of the current value in the CLKOUT bit.</p> <p>User and privilege mode (read):</p> <p>0 The pin is configured to output a logic low (0) (output voltage is <math>V_{OL}</math> or lower).</p> <p>1 The pin is configured to output logic high (1) (output voltage is <math>V_{OH}</math> or higher).</p> <p>User and privilege mode (write):</p> <p>0 Writing a zero to this bit has no effect.</p> <p>1 Set pin to logic high (1) (output voltage is <math>V_{OH}</math> or higher).</p>
16	SYNCSET	0 1	<p><b>Sets output state of RTPSYNC pin logic high.</b> Value in the SYNCSET bit sets the data output control register bit to 1 regardless of the current value in the SYNCOUT bit.</p> <p>User and privilege mode (read):</p> <p>0 The pin is configured to output a logic low (0) (output voltage is <math>V_{OL}</math> or lower).</p> <p>1 The pin is configured to output logic high (1) (output voltage is <math>V_{OH}</math> or higher).</p> <p>User and privilege mode (write):</p> <p>0 Writing a zero to this bit has no effect.</p> <p>1 Set pin to logic high (1) (output voltage is <math>V_{OH}</math> or higher).</p>
15-0	DATASET	0 1	<p><b>Sets output state of RTPDATA[15:0] pins to logic high.</b> Value in the DATAxSET bit sets the data output control register bit to 1 regardless of the current value in the DATAxOUT bit.</p> <p>User and privilege mode (read):</p> <p>0 The pin is configured to output a logic low (0) (output voltage is <math>V_{OL}</math> or lower).</p> <p>1 The pin is configured to output logic high (1) (output voltage is <math>V_{OH}</math> or higher).</p> <p>User and privilege mode (write):</p> <p>0 Writing a zero to this bit has no effect.</p> <p>1 Set pin to logic high (1) (output voltage is <math>V_{OH}</math> or higher).</p>

### 30.4.13 RTP Pin Control 5 Register (RTPPC5)

This register provides the option to set pins to a logic 0 level without influencing the state of other pins. It eliminates the read-modify-write operation necessary with RTPPC3 (Section 30.4.11). Once the pin is configured in functional mode (RTPPC0, Section 30.4.8), setting the corresponding bit to one in RTPPC5 will not affect the pinstate. Figure 30-21 and Table 30-21 illustrate this register.

**Figure 30-21. RTP Pin Control 5 Register (RTPPC5) [offset = 48h]**

31	Reserved	19	18	17	16
			ENACL R	CLKCL R	SYNCL R
R-0			R/W-0	R/W-0	R/W-0
15	DATACLR				0
R/W-0					

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 30-21. RTP Pin Control 5 Register (RTPPC5) [offset = 48h] Field Descriptions**

Bit	Field	Value	Description
31-19	Reserved	0	Read returns 0. Writes have no effect.
18	ENACL	0	<b>Sets the output state of RTPENA pin to logic high.</b> Value in the ENASET bit sets the data output control register bit to 1 regardless of the current value in the ENAOUT bit User and privilege mode (read): The pin is configured to output a logic low (0) (output voltage is $V_{OL}$ or lower).
		1	The pin is configured to output logic high (1) (output voltage is $V_{OH}$ or higher).
		0	User and privilege mode (write): Writing a zero to this bit has no effect.
		1	Set pin to logic low (0) (output voltage is $V_{OL}$ or lower).
17	CLKCLR	0	<b>Sets output state of RTPCLK pin to logic low.</b> Value in the CLKCLR bit sets the data output control register bit to 0 regardless of the current value in the CLKOUT bit User and privilege mode (read): The pin is configured to output a logic low (0) (output voltage is $V_{OL}$ or lower).
		1	The pin is configured to output logic high (1) (output voltage is $V_{OH}$ or higher).
		0	User and privilege mode (write): Writing a zero to this bit has no effect.
		1	Set pin to logic low (0) (output voltage is $V_{OL}$ or lower).
16	SYNCLR	0	<b>Sets output state of RTPSYNC pin logic low.</b> Value in the SYNCLR bit clears the data output control register bit to 0 regardless of the current value in the SYNCOUT bit. User and privilege mode (read): The pin is configured to output a logic low (0) (output voltage is $V_{OL}$ or lower).
		1	The pin is configured to output logic high (1) (output voltage is $V_{OH}$ or higher).
		0	User and privilege mode (write): Writing a zero to this bit has no effect.
		1	Set pin to logic low (0) (output voltage is $V_{OL}$ or lower).
15-0	DATACLR	0	<b>Sets output state of RTPDATA[15:0] pins to logic low.</b> Value in the DATAxCLR bit clears the data output control register bit to 0 regardless of the current value in the DATAxOUT bit. User and privilege mode (read): The pin is configured to output a logic low (0) (output voltage is $V_{OL}$ or lower).
		1	The pin is configured to output logic high (1) (output voltage is $V_{OH}$ or higher).
		0	User and privilege mode (write): Writing a zero to this bit has no effect.
		1	Set pin to logic low (0) (output voltage is $V_{OL}$ or lower).

### 30.4.14 RTP Pin Control 6 Register (RTPPC6)

These bits configure the pins in push-pull or open-drain functionality. If configured to be open-drain, the module only drives a logic low level on the pin. An external pull-up resistor needs to be connected to the pin to pull it high when the pin is in high-impedance mode. [Figure 30-22](#) and [Table 30-22](#) illustrate this register.

**Figure 30-22. RTP Pin Control 6 Register (RTPPC6) [offset = 4Ch]**

31	Reserved	19	18	17	16
			ENAPDR	CLKPDR R	SYNCPDR
R-0			R/W-0	R/W-0	R/W-0
15	DATAPDR				0
R/W-0					

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 30-22. RTP Pin Control 6 Register (RTPPC6) Field Descriptions**

Bit	Field	Value	Description
31-19	Reserved	0	Read returns 0. Writes have no effect.
18	ENAPDR	0 1	<p><b>RTPEN<math>\bar{A}</math> Open drain enable.</b> This bit enables open drain functionality on the pin if it is configured as a GIO output (RTPPC0[18]=0; RTPPC1[18]=1). If the pin is configured as a functional pin (RTPPC0[18]=1), the open drain functionality is disabled.</p> <p>User and privilege mode (read):</p> <p>0 Pin behaves as normal push/pull pin</p> <p>1 Pin operates in open drain mode</p> <p>User and privilege mode (write):</p> <p>0 Configures the pin as push/pull</p> <p>1 Configures the pin as open drain</p>
17	CLKPDR	0 1	<p><b>RTPCLK Open drain enable.</b> This bit enables open drain functionality on the pin if it is configured as GIO output (RTPPC0[17]=0; RTPPC1[17]=1). If the pin is configured as functional pin (RTPPC0[17]=1), the open drain functionality is disabled.</p> <p>User and privilege mode (read):</p> <p>0 Pin behaves as normal push/pull pin</p> <p>1 Pin operates in open drain mode</p> <p>User and privilege mode (write):</p> <p>0 Configures the pin as push/pull</p> <p>1 Configures the pin as open drain</p>
16	SYNCPDR	0 1	<p><b>RTPSYNC Open drain enable.</b> This bit enables open drain functionality on the pin if it is configured as a GIO output (RTPPC0[16]=0; RTPPC1[16]=1). If pin is configured as functional pin (RTPPC0[16]=1), the open drain functionality is disabled.</p> <p>User and privilege mode (read):</p> <p>0 Pin behaves as normal push/pull pin</p> <p>1 Pin operates in open drain mode</p> <p>User and privilege mode (write):</p> <p>0 Configures the pin as push/pull</p> <p>1 Configures the pin as open drain</p>

**Table 30-22. RTP Pin Control 6 Register (RTPPC6) Field Descriptions (continued)**

Bit	Field	Value	Description
15-0	DATAPDR		<p><b>RTPDATA[15:0] Open drain enable.</b> These bits enable open drain functionality on the pins if they are configured as a GIO output (RTPPC0[15:0]=0; RTPPC1[15:0]=1). If the pins are configured as a functional pins (RTPPC0[15:0]=1), the open drain functionality is disabled.</p> <p>User and privilege mode (read):</p> <p>0 Pin behaves as normal push/pull pin</p> <p>1 Pin operates in open drain mode</p> <p>User and privilege mode (write):</p> <p>0 Configures the pin as push/pull</p> <p>1 Configures the pin as open drain</p>

### 30.4.15 RTP Pin Control 7 Register (RTPPC7)

The bits in register control the pullup/down functionality of a pin. The internal pullup/down can be enabled or disabled by this register. The reset configuration of these bits is device implementation dependent. Please consult the device datasheet this information. [Figure 30-23](#) and [Table 30-23](#) illustrate this register.

**Figure 30-23. RTP Pin Control 7 Register (RTPPC7) [offset = 50h]**

31	19	18	17	16
Reserved		ENADIS	CLKDIS	SYNCDIS
R-0		R/W-x	R/W-x	R/W-x
15	DATADIS			0
R/W-x				

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 30-23. RTP Pin Control 7 Register (RTPPC7) Field Descriptions**

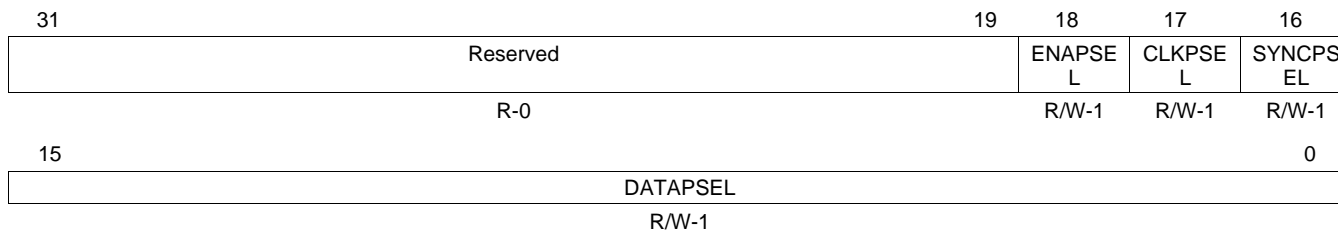
Bit	Field	Value	Description
31-19	Reserved	0	Read returns 0. Writes have no effect.
18	ENADIS		<b>RTPEN<math>\bar{A}</math> Pull disable.</b> This bit removes internal pullup/pulldown functionality from the pin when it is configured as an input pin (RTPPC1[18]=0).
		0	User and privilege mode (read): Pullup/pulldown functionality enabled
		1	Pullup/pulldown functionality disabled
			User and privilege mode (write): 0 Enables pullup/pulldown functionality 1 Disables pullup/pulldown functionality
17	CLKDIS		<b>RTPCLK Pull disable.</b> This bit removes the internal pullup/pulldown functionality from the pin when it is configured as an input pin (RTPPC1[17]=0).
		0	User and privilege mode (read): Pullup/pulldown functionality enabled
		1	Pullup/pulldown functionality disabled
			User and privilege mode (write): 0 Enables pullup/pulldown functionality 1 Disables pullup/pulldown functionality
16	SYNCDIS		<b>RTPSYNC Pull disable.</b> Removes internal pullup/pulldown functionality from the pin when configured as an input pin (RTPPC1[16]=0).
		0	User and privilege mode (read): Pullup/pulldown functionality enabled
		1	Pullup/pulldown functionality disabled
			User and privilege mode (write): 0 Enables pullup/pulldown functionality 1 Disables pullup/pulldown functionality
15-0	DATADIS		<b>RTPDATA[15:0 ] Pull disable.</b> Removes internal pullup/pulldown functionality from the pins when configured as input pins (RTPPC1[15:0]=0).
		0	User and privilege mode (read): Pullup/pulldown functionality enabled
		1	Pullup/pulldown functionality disabled
			User and privilege mode (write): 0 Enables pullup/pulldown functionality 1 Disables pullup/pulldown functionality



### 30.4.16 RTP Pin Control 8 Register (RTPPC8)

These bits control if the internal pullup or pulldown is configured on the input pin. A secondary function exists when the pull configuration is disabled and a pulldown is selected. This will disable the input buffer. Figure 30-24 and Table 30-24 illustrate this register.

**Figure 30-24. RTP Pin Control 8 Register (RTPPC8) [offset = 54h]**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 30-24. RTP Pin Control 8 Register (RTPPC8) [offset = 54h] Field Descriptions**

Bit	Field	Value	Description
31-19	Reserved	0	Read returns 0. Writes have no effect.
18	ENAPSEL	0	<b>RTPEN<math>\bar{A}</math> Pull select.</b> This bit configures pullup or pulldown functionality if RTPPC7[18]=0. User and privilege mode (read): Pulldown functionality enabled
		1	Pullup functionality enabled
		0	User and privilege mode (write): Enables pulldown functionality
		1	Enables pullup functionality
17	CLKPSEL	0	<b>RTPCLK Pull select.</b> This bit configures pullup or pulldown functionality if RTPPC7[17]=0. User and privilege mode (read): Pulldown functionality enabled
		1	Pullup functionality enabled
		0	User and privilege mode (write): Enables pulldown functionality
		1	Enables pullup functionality
16	SYNCPSEL	0	<b>RTPSYNC Pull select.</b> This bit configures pullup or pulldown functionality if RTPPC7[16]=0. User and privilege mode (read): Pulldown functionality enabled
		1	Pullup functionality enabled
		0	User and privilege mode (write): Enables pulldown functionality
		1	Enables pullup functionality
15-0	DATAPSEL	0	<b>RTPDATA[15:0] Pull select.</b> These bits configure pullup or pulldown functionality if RTPPC7[15:0]=0. User and privilege mode (read): Pulldown functionality enabled
		1	Pullup functionality enabled
		0	User and privilege mode (write): Enables pulldown functionality
		1	Enables pullup functionality

---

**NOTE:** If the pullup/down is disabled in RTPPC7 and configured as pulldown in RTPPC8, then the input buffer is disabled.

---

## IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI 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 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. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

### Products

Audio	<a href="http://www.ti.com/audio">www.ti.com/audio</a>
Amplifiers	<a href="http://amplifier.ti.com">amplifier.ti.com</a>
Data Converters	<a href="http://dataconverter.ti.com">dataconverter.ti.com</a>
DLP® Products	<a href="http://www.dlp.com">www.dlp.com</a>
DSP	<a href="http://dsp.ti.com">dsp.ti.com</a>
Clocks and Timers	<a href="http://www.ti.com/clocks">www.ti.com/clocks</a>
Interface	<a href="http://interface.ti.com">interface.ti.com</a>
Logic	<a href="http://logic.ti.com">logic.ti.com</a>
Power Mgmt	<a href="http://power.ti.com">power.ti.com</a>
Microcontrollers	<a href="http://microcontroller.ti.com">microcontroller.ti.com</a>
RFID	<a href="http://www.ti-rfid.com">www.ti-rfid.com</a>
OMAP Mobile Processors	<a href="http://www.ti.com/omap">www.ti.com/omap</a>
Wireless Connectivity	<a href="http://www.ti.com/wirelessconnectivity">www.ti.com/wirelessconnectivity</a>

### Applications

Communications and Telecom	<a href="http://www.ti.com/communications">www.ti.com/communications</a>
Computers and Peripherals	<a href="http://www.ti.com/computers">www.ti.com/computers</a>
Consumer Electronics	<a href="http://www.ti.com/consumer-apps">www.ti.com/consumer-apps</a>
Energy and Lighting	<a href="http://www.ti.com/energy">www.ti.com/energy</a>
Industrial	<a href="http://www.ti.com/industrial">www.ti.com/industrial</a>
Medical	<a href="http://www.ti.com/medical">www.ti.com/medical</a>
Security	<a href="http://www.ti.com/security">www.ti.com/security</a>
Space, Avionics and Defense	<a href="http://www.ti.com/space-avionics-defense">www.ti.com/space-avionics-defense</a>
Transportation and Automotive	<a href="http://www.ti.com/automotive">www.ti.com/automotive</a>
Video and Imaging	<a href="http://www.ti.com/video">www.ti.com/video</a>

TI E2E Community Home Page

[e2e.ti.com](http://e2e.ti.com)

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265  
Copyright © 2011, Texas Instruments Incorporated